XMC1000, XMC4000

32-bit Microcontroller Series for Industrial Applications

Event Request Unit (ERU)

AP32306

Application Note

About this document

Scope and purpose

This application note describes the Event Request Unit (ERU) available in the XMC1000 and XMC4000 microcontroller families.

Intended audience

This document is intended for engineers who have a good understanding of the XMC1000 and XMC4000 microcontrollers.

Applicable Products

- XMC1000 and XMC4000 Microcontroller Families

References

Infineon: Example Code, http://www.infineon.com/XMC1000 Tab: Documents

Infineon: XMC Lib, http://www.infineon.com/DAVE

Infineon: DAVE™, http://www.infineon.com/DAVE

Infineon: XMC Reference Manual, http://www.infineon.com/XMC1000 (or /XMC4000) Tab: Documents

Infineon: XMCData sheet, http://www.infineon.com/XMC1000 (or /XMC4000) Tab: Documents

Table of Contents

Table of Contents

1 Basics

The Event Request Unit (ERU) enables time-critical interconnections when total real-time correctness and safety is required. The ERU has tailor-made event schemes that provide high-level control interactions between different modules. This reduces SW in timer-ADC-IO interactions.

1.1 The Concept

The Event Request Unit (ERU) is a modular logic that can select, combine, detect and memorize peripheral events. When the gating conditions are met, the ERU can generate trigger pulses that are distributed back to the system as requests for HW actions. The ERU can select input signals from up to 32 event sources and request actions through four output channels.

The ability of the ERU to control events enables the creation of comprehensive embedded tasks in HW. The event flow and action scheme can be configured by the user and then the ERU can process actions from other parts of the system without any ongoing SW control.

1.2 Use Cases

The ERU is typically used to provide an alternative signal path when there is no direct signal path available. For example, the ERU can signal the start of ADC conversions based on specific conditions like a port pin state, a time window due to a second timer, or a certain event pattern.

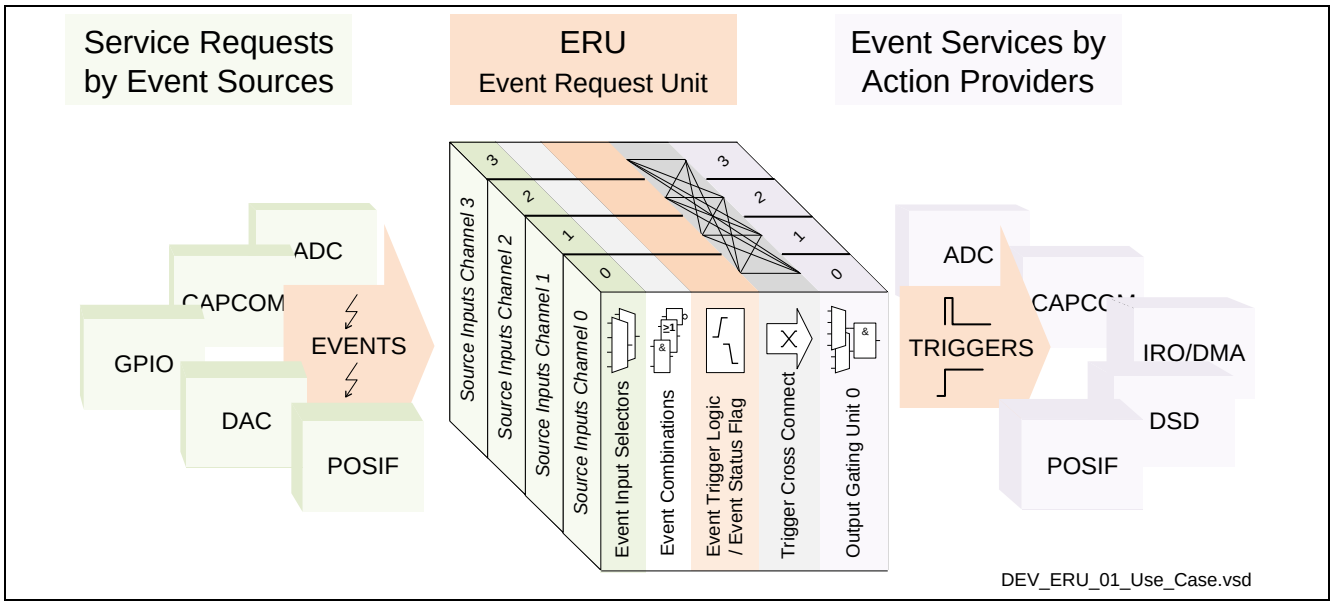

Figure 1 Use Cases for an Event Request Unit (ERU)

2 Implementations

In the XMC4000 microcontrollers there are two ERUs: ERU0 and ERU1. The two ERUs are functionally the same but each has a different application focus due to the nature of their connections to the system (see Figure 2).

Implementations

- ERU0 selects External Event Requests, mainly from General Purpose IOs (GPIO).
- ERU1 selects Internal Event Requests from embedded peripherals (PERIPH) and also from General Purpose IOs.

ERU0 links trigger pulses (named TRIGGER) to the interrupt system only. ERU1 can also link trigger pulses to the peripherals and distribute Event Pattern Match Detect status signals (named LEVEL).

Figure 2 shows how units are interfaced to the ERUs and interact via a Top-Level Interconnect Matrix. To complete the picture of possible interaction scenarios, the diagram also shows how operations can be extended to involve DMA transfers (by the GPDMA) which are triggered by a handler (DLR) on the Interrupt Service Request Lines (SRn).

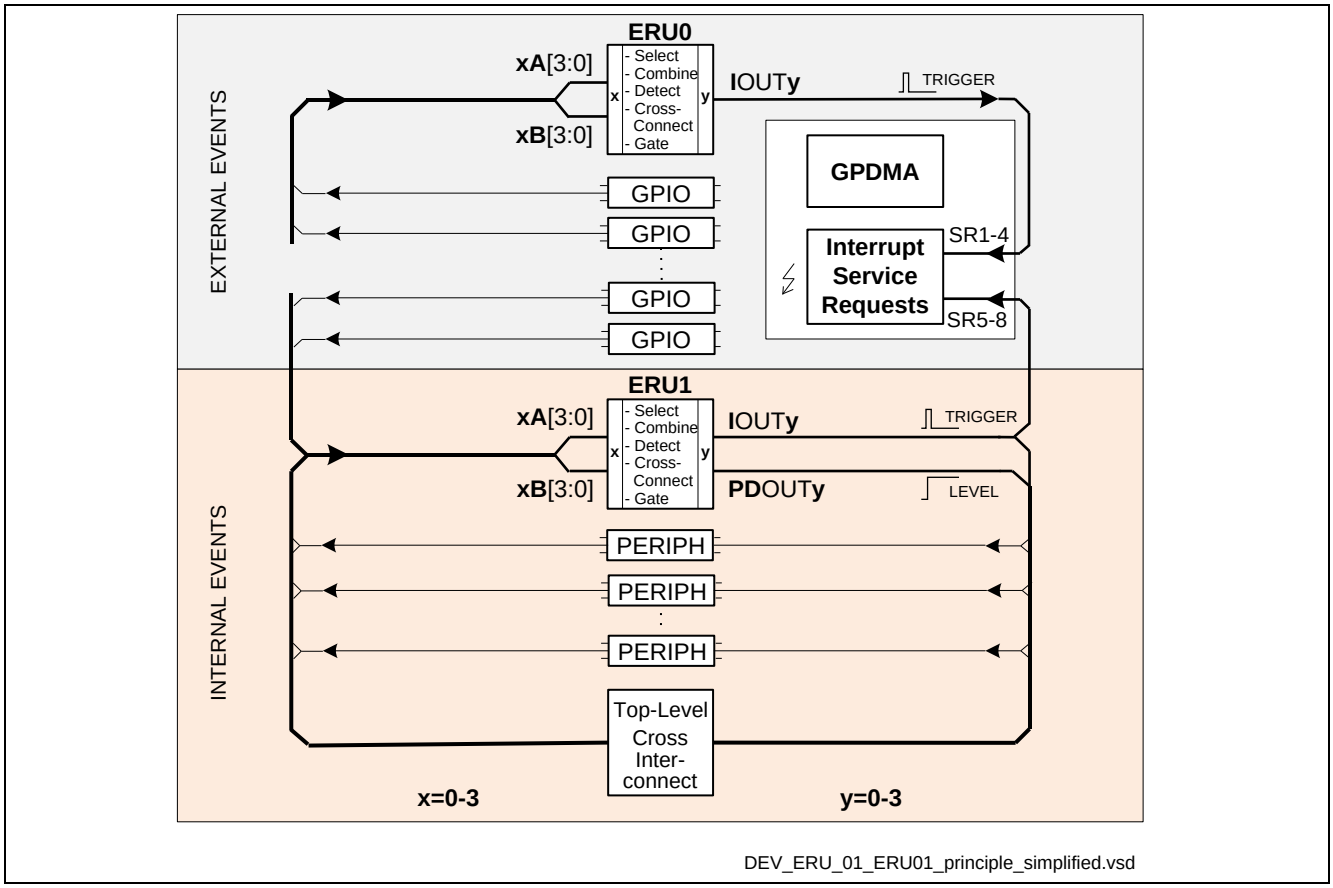

Figure 2 Signal Flow Principle with ERUs in XMC4000

In the XMC1000 microcontroller, up to two ERUs are implemented: ERU0 and ERU1. Unlike the XMC4000, the ERUs in the XMC1000 are not differentiated by the connections they have to the system (see Figure 3). Both ERUs (ERUx) select Internal Event Requests from both embedded peripherals (PERIPH) and General Purpose IOs. ERUx links the trigger pulses (named TRIGGER) to the interrupt system and the peripherals. Both ERUs can distribute Event Pattern Match Detect status signals (named LEVEL).

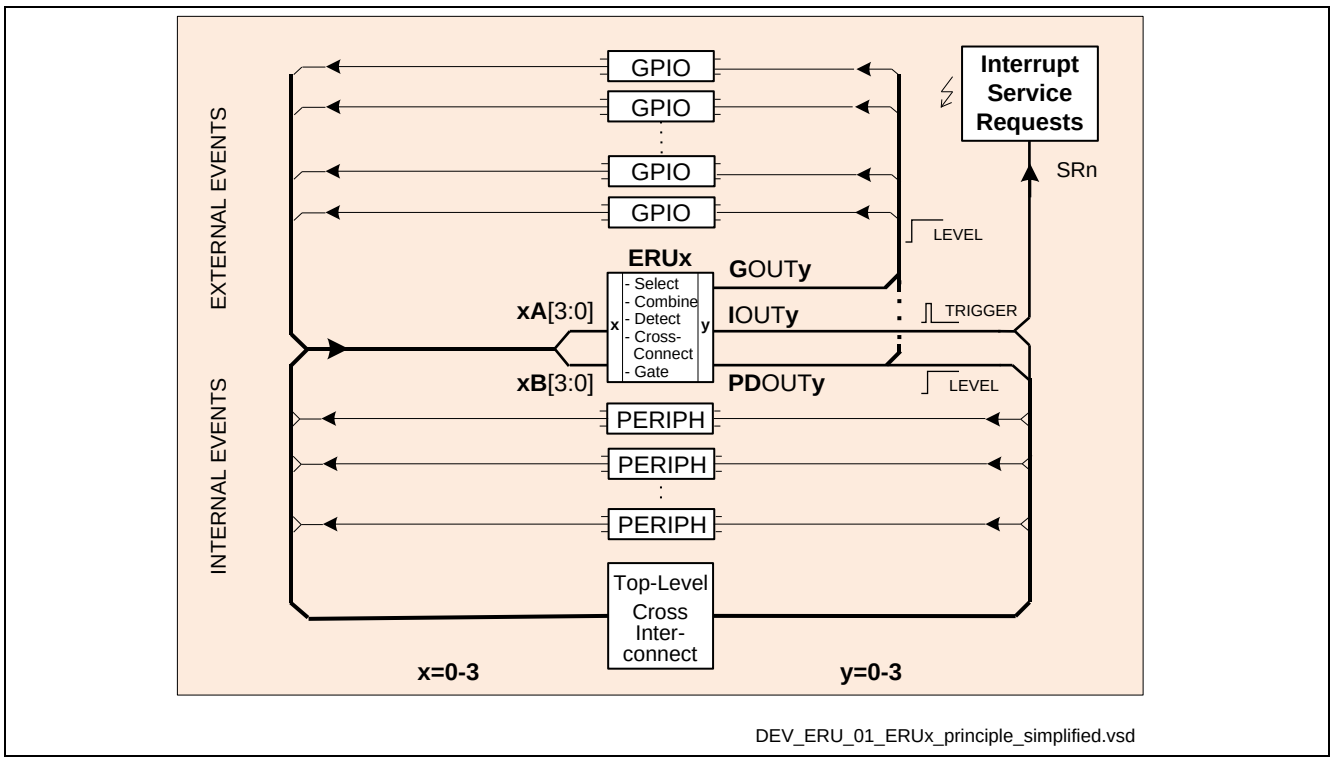

Figure 3 Signal Flow Principle with ERUs in XMC1000

Note: The functional description and examples provided in this document are based on the XMC4000 implementation unless otherwise stated. Nevertheless, besides the differences in the ERU port connections and the lack of support for DMA transfers, they can be mostly applied also to the XMC1000.

2.1 Event Request Selection

Input lines, xA(3:0) and xB(3:0), are connected to the corresponding ERU input channel x (x=0-3). Each input line of the ERU is hard-wired to a specific event source. A specific Event Request, from an I/O pin or a peripheral unit, is selected by a MUX switch at either the xA(3:0) or at the xB(3:0) input line group of ERU0 or ERU1, according to the ERU Pin Connection tables.

2.2 Signal Combination Logic

An Event Request signal pair (A,B), selected by the input line groups xA(3:0) and xB(3:0) respectively, can be processed as a compound event. The Combination Logic of the channel x input stage can create a conditional signal before it is transferred to the Event Trigger Logic stage for event edge detection.

2.3 Input and Trigger Functions

A signal edge is regarded as a true event edge if it complies with the considered edge, positive or negative. If it is a true event, an event flag FLx is set, a trigger pulse TRx is created and, if enabled, the event and the event flag level status are passed to the Cross Connect Select stage. The flag can be reset by a false event or by SW.

2.4 Cross Connect Selection

The Cross Connect Select stage, for each channel x of an ERU, is an Event Trigger pulse routing matrix that is able to switch the trigger pulse distribution path back to the system via any output channel y (y=0-3). This can target a certain event service action provider according to the Top-Level control by the ERU Pin Connections tables.

2.5 Output Gating Selection

The Output Gating Selection stage of an ERU can gate the Trigger Pulse signals (TRIGGER) in the output channels IOUTy (y=0-3) by the Event Pattern Match/Mismatch Detect status (LEVEL) output signal, PDOUTy (y=0-3). An Event Pattern is a combination of memorized x-channel events stored in the event flags FLx (x=0-3).

2.5.1 Output Gating by Designated Peripheral Trigger Inputs

ERU1 has designated trigger inputs from the ADC and from some of the CAPCOM4/-8 units that can be linked directly to the Gated Trigger stage of an Output Gating Unit channel (OGUy). By this “kitchen entrance”-like concept, these Peripheral Triggers act immediately, unconditionally in the output gating by the event flags FLx.

3 Top-Level Interconnect Matrix

Most interconnect lines go directly from one system unit to another system unit without passing through an ERU. For example, an ADC conversion can be started directly by a signal from a timer. These immediate interconnections are described virtually by the so called Top-Level Interconnect matrix – a table which is embodied by a block in the drawings (see Figure 4).

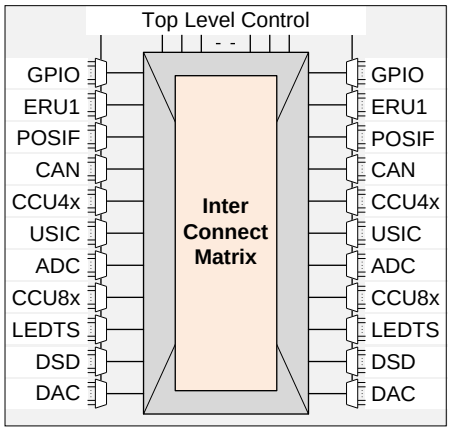

Figure 4 The Top-Level Interconnect Matrix

3.1 The ERU Pin Connection Tables

The ERU Pin Connection tables (which describe the ERU0 and ERU1 interfaces to peripheral units or IOs) are actually subsets of the total system Top-Level Interconnect matrix – split up like this in order to ease understanding of those paths where an ERU is involved.

3.2 The Principle Blocks of the ERU0 vs. the ERU1 in Detail

The principle difference between the ERU0 and the ERU1 implementations in XMC4000 are which event request sources they are connected to as service providers and which action providers they are linked to as service requestors: ERU0 for external events - and ERU1 for internal events plus 3 Peripheral Trigger “kitchen entrances” (see Figure 5).

Top-Level Interconnect Matrix

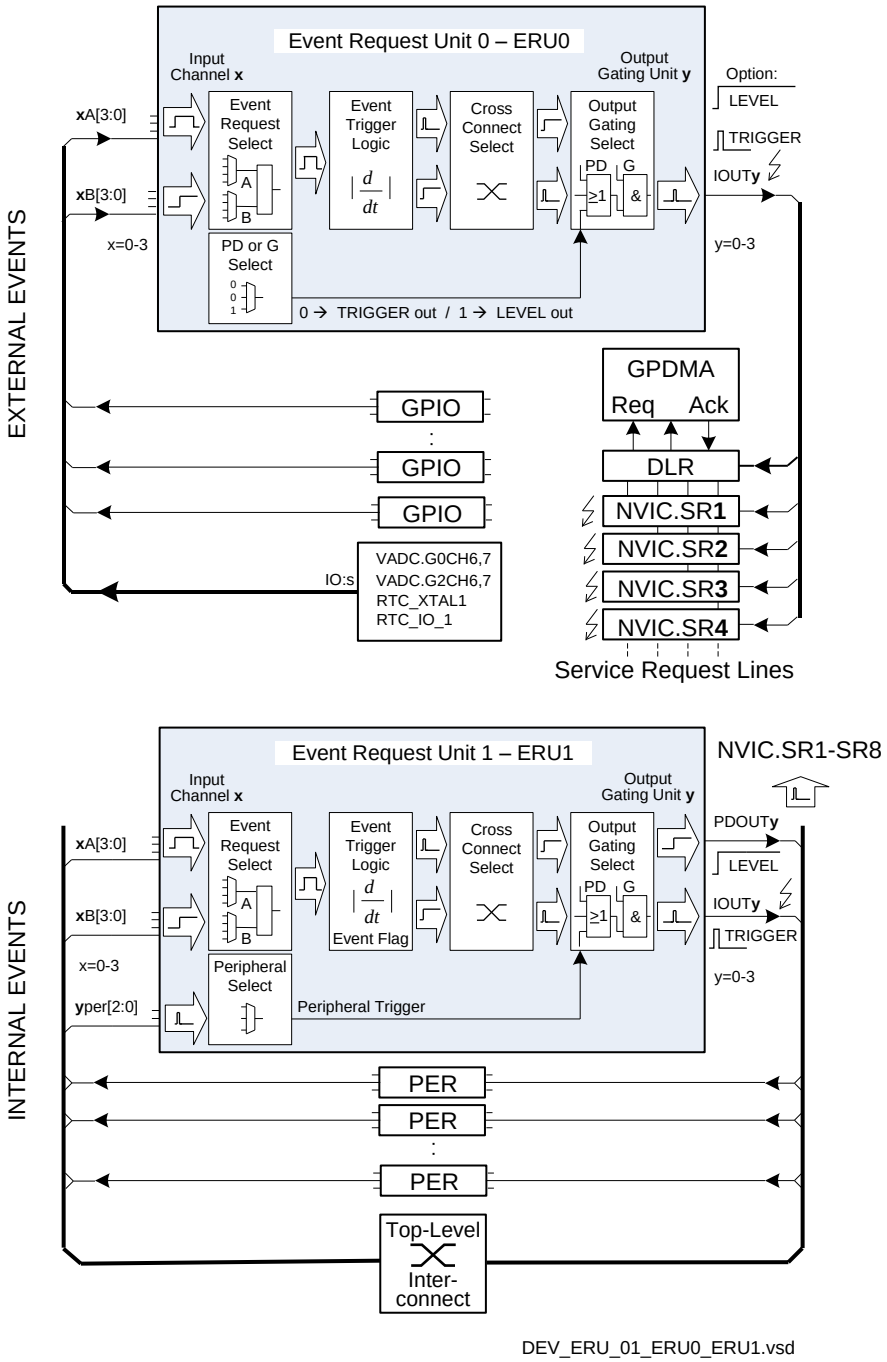

Figure 5 XMC4000 Event Request Unit Overview (ERU0 versus ERU1)

4 Getting Started with Event Request Unit – ERU

There is one Input Selection Register and two Control Registers per input channel x (x=0-3). These are used to setup the entire functionality of an ERU at the Top Level and to target the Output Channels y (y=0-3):

- EXISEL Input Selection Register
- EXICONx Input and Trigger Control Register
- EXOCONx Output Control Register

Initializations and functions are determined by writing to the respective register bitfields (see Figure 6).

4.1 Initialization Example

```
// Set up ERU1 Input Channel x=0, targeting the Output Channel y=2 (IOUT2):

// Select Event Request for input A and B in the ERU1_EXISEL register bitfields:
EXS0A = 0; //select input ERU1_0A0 – i.e. A connected to GPIO port pin P1.5
EXS0B = 1; //select input ERU1_0B1 – i.e. B connected to CCU80.ST0 status bit 0

// Select logical combinations that should be taken into account as event request
NA = 0; // input A is used directly, i.e. not inverted
NB = 0; // input A is used directly, i.e. not inverted
SS = 3; // setup source combination, i.e. logical condition: input A AND input B

// Select Event Trigger Logic conditions for trigger, level detect and event edge
PE = 1; // set trigger pulse enable, i.e. an output trigger pulse will be created
LD = 0; // define event flag sticky, i.e. it will not be rebuild by HW, but by SW
RE = 1; // detection on Rising Edge, i.e. event edge on positive signal transition
FE = 0; // no detection on Falling Edge

// Perform Cross Connect Selection, i.e. select target output channel y (here y=2)
// for the Event Trigger Logic Output Pulse TR0 from input channel x (here x=0):
OCS = 2; // the channel output (IOUT2) would trigger the following destinations:
     // CCU4x.IN2(K)  - i.e. CAPCOM4 Unit CCU4x, Slice CC42 Timer Input
     // CCU8x.IN2(G)  - i.e. CAPCOM8 Unit CCU8x, Slice CC42 Timer Input
     // VADC.G2REQTRN- i.e. ADC Trigger Request N Input, Group 2
     // VADC.G3REQTRN- i.e. ADC Trigger Request N Input, Group 3
     // ERU1.1B3 - i.e. A trigger feed-back to ERU1 Channel Input 1B3
     // NVIC.SR7 - i.e. A trigger puls to Service Request Line nr 7
     // POSIF0.MSET(F) - i.e. POSIF 0 Multi Channel Next Pattern Update Set
     // POSIF1.MSET(F)- i.e. POSIF 1 Multi Channel Next Pattern Update Set
```

```
// Select Event Output Trigger Control 2 Register ERU1_EXOCON2 for output gating
// Select Output Gating functions – Here: just a straight forward alternative:
ISS = 0;  // set the Internal Peripheral Trigger Source Selection =0, “no source”
GEEN = 0; // disable “Gating Event Enable on Pattern Detection Changes” trigger
GP = 1;   // set “Output Gating Select on Pattern Detection” =1, to activate IOUT2
IPEN0, IPEN1, IPEN2, IPEN3 = 0; // disable the Pattern Detection Enable flags
```

4.2 Event Request Unit Block Diagrams and Control by Registers

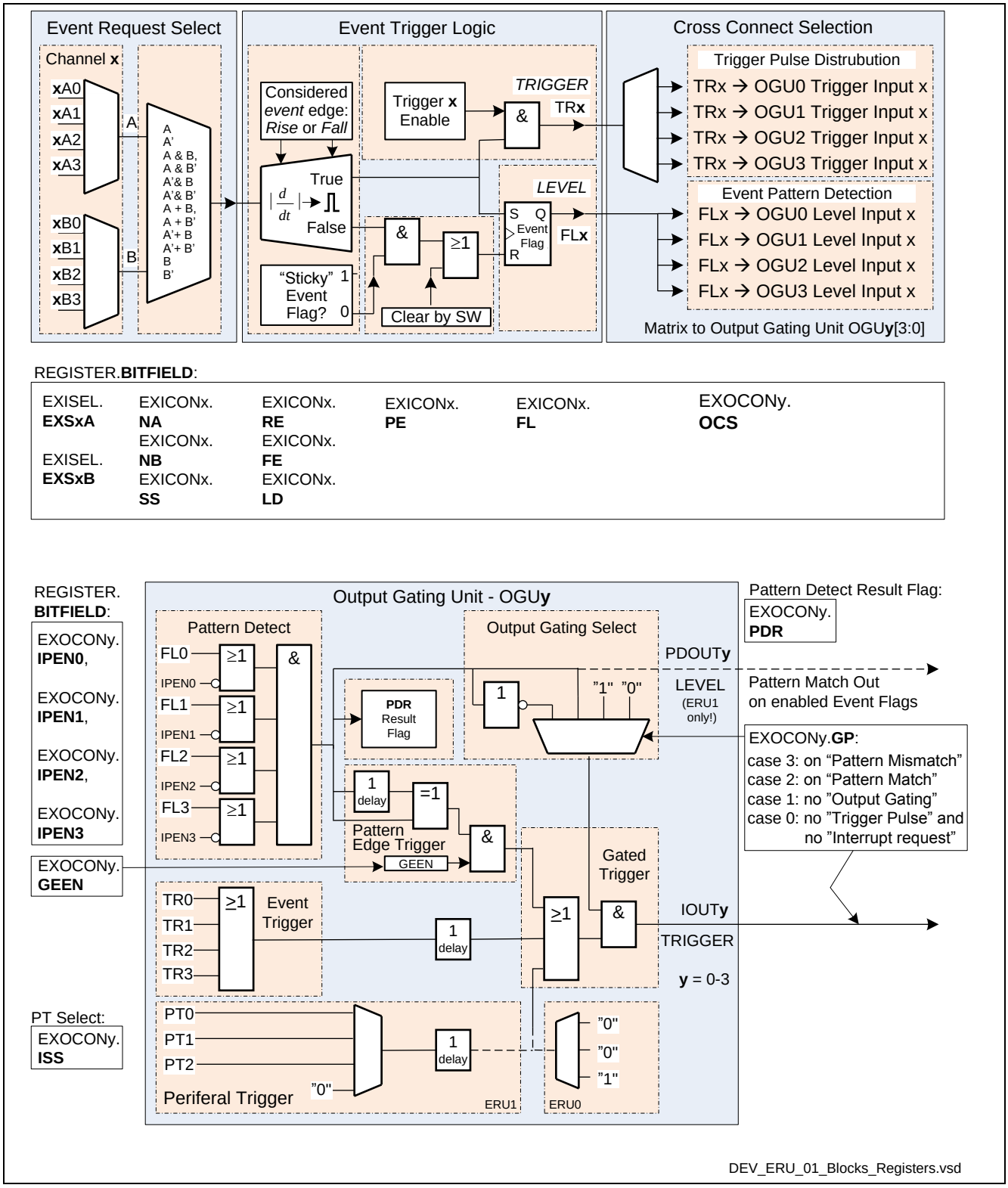

Figure 6 Event Request Unit Block Diagrams and Control by Registers

5 Top-Level Control of External Event Requests

Instead of the traditional concept of External Interrupts, the notion here is rather of External Events. This is because the device can do more than just enabling external interrupt services.

5.1 External Interrupts as Events

Service request signals from external sources are always regarded as event requests and may benefit from the value adding logic operations offered by the ERU. For example, the service request can be routed on-the-fly by cross connection within the ERU to alternative service providers or immediate DMA support. All this is achieved without any latency impact.

5.2 Handling External Service Requests by the ERU

The Event Request Unit 0 (ERU0) selects and detects external events requests. It then creates and routes each Trigger Pulse (TRIGGER) to the appropriate service provider via a Service Request Line. This process addresses a service provider in SW, or by extension, a DMA action (see Figure 7).

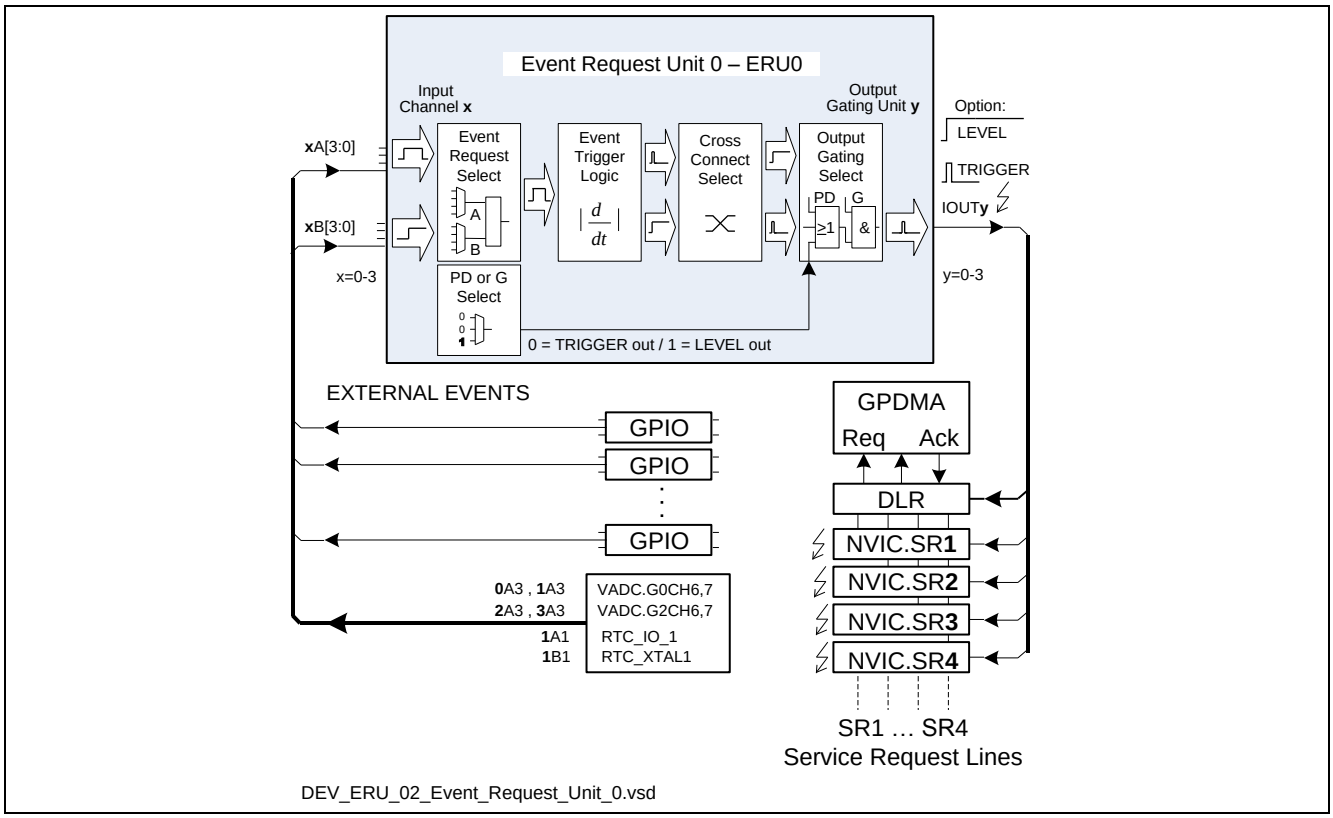

Figure 7 Schematic view of the ERU0 blocks for handling External Event Requests

5.3 Scope of the ERU0 Event Request Sources

The Event Request Unit 0 (ERU0) has no internal event request sources. It only handles external event requests coming from off-chip sources, via General Purpose IO Port Pins (GPIO). However, some special IOs can be found, namely the dedicated IO-pins for Voltage Range Comparators and RTC. See Table 1.

Table 1 ERU0 Pin Connections

Global Inputs	Connected To	Corresponding Port Pins respectively
ERU0_0A0 … ERU0_0A2	GPIO	P0.1, P3.2, P2.5
ERU0_0A3	VADC_G0CH6	P14.6 (used for out of range comparator)
ERU0_0B0 … ERU0_0B3	GPIO	P0.0, P3.1, P2.4
ERU0_1A0	GPIO	P0.10,
ERU0_1A1A, ERU0_1A1B	RTC_IO_1	P3.7
ERU0_1A2	GPIO	P2.3
ERU0_1A3	VADC_G0CH7	P14.7 (used for out of range comparator)
ERU0_1B0	GPIO	P0.9
ERU0_1B1	RTC_XTAL1	P3.7
ERU0_1B2 … ERU0_1B3	GPIO	P2.2, P2.6
ERU0_2A0 … ERU0_2A2	GPIO	P1.5, P0.8, P0.13
ERU0_2A3	VADC_G2CH6	P15.6 (used for out of range comparator)
ERU0_2B0 … ERU0_2B3	GPIO	P1.4, P0.7, P0.12, P0.4
ERU0_3A0 … ERU0_3A2	GPIO	P1.1, P3.6, P0.11
ERU0_3A3	VADC_G2CH7	P15.7 (used for out of range comparator)
ERU0_3B0 … ERU0_3B3	GPIO	P1.0, P3.5, P0.6, P0.2

5.4 Service Request Handling

From the concept point of view an ERU is both a service provider and a service request source. ERU0 is the only input for all external event requests and thus a service provider. In turn ERU0 is a service request source for the NVIC (i.e. the Nested Vector Interrupt Controller) or the DMA Handler (DLR) for the DMA support.

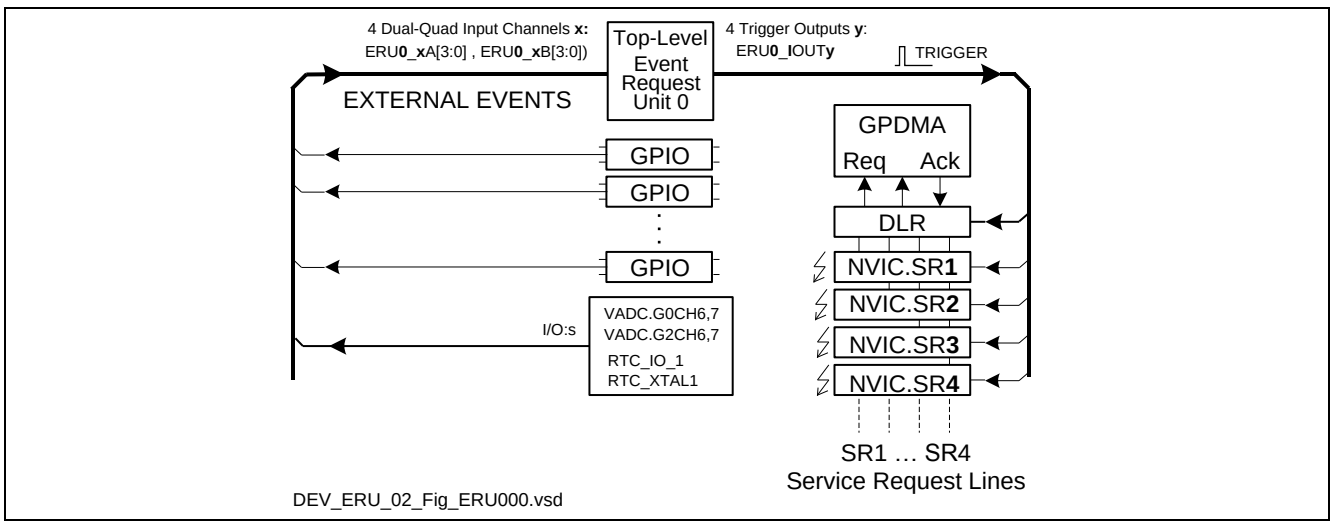

Figure 8 Signal Flow Principle with the Event Request Unit 0 – ERU0

5.5 Examples of Versatile External Interrupt Request Scenarios

Figure 9 shows the four fundamental stages of routing external requests via ERU0 to the service providers. Any mix of using these fundamental ERU0 operations shown by the scenarios Ex1 - Ex4 can be performed as well. Note the pattern detection condition of two consecutive events in Ex4 that performs a state-machine-like task.

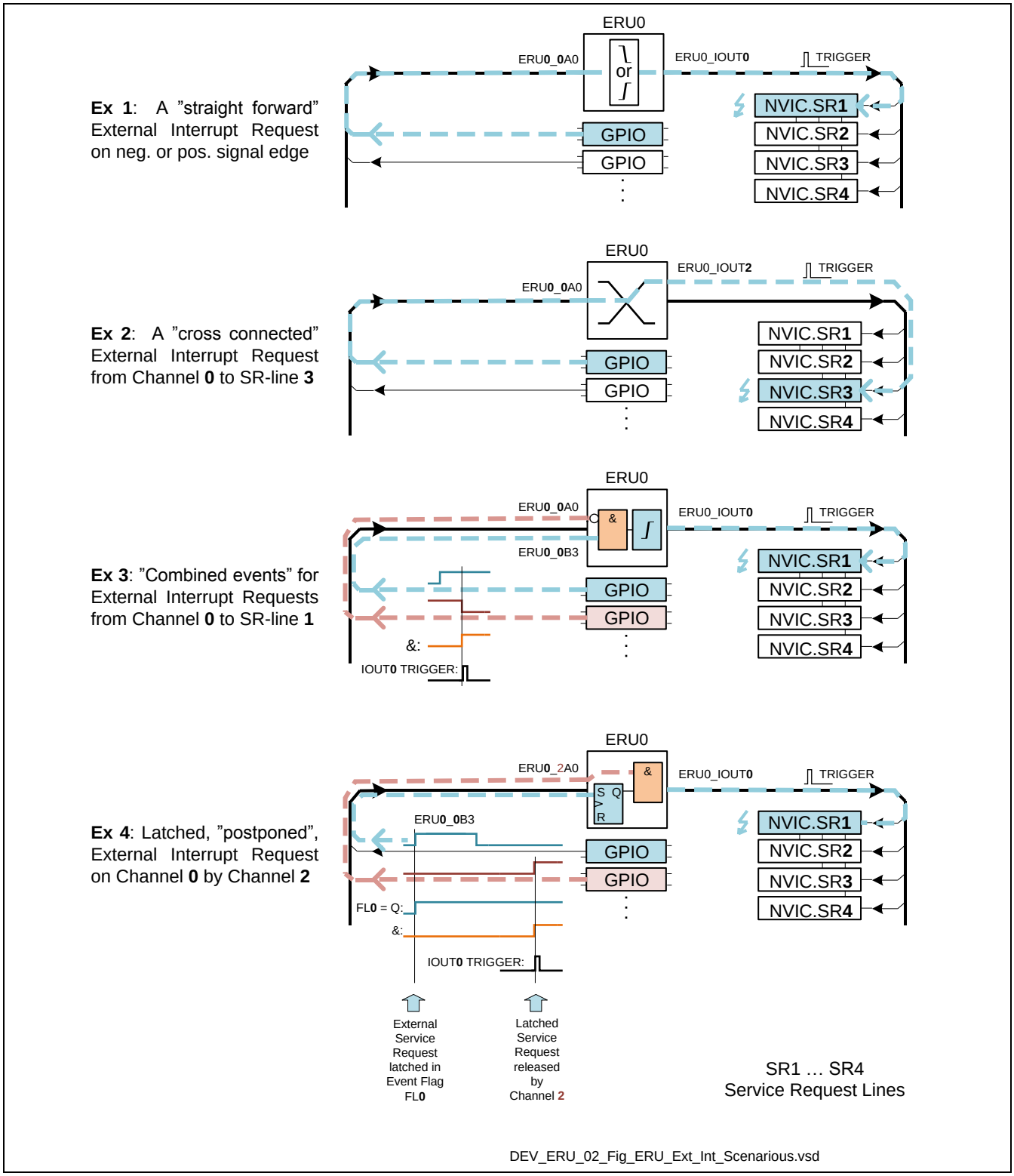

Figure 9 Examples of Versatile External Interrupt Request Scenarios with the ERU0

5.6 Use Case Example 1: Triggering an External Interrupt Request

In this use case example, ERU0 is used to trigger an external interrupt request upon detection of a falling edge on the port pin P2.5. When the interrupt is serviced by the CPU, the port pin P0.0 is toggled.

The example is developed based on the XMC1200 device.

5.6.1 XMC Lib Implementation

The next sections describe how the Infineon XMC library (XMC Lib) can be used to implement the example.

5.6.1.1 Configuration

P2.5 is connected to the channel 1 input A1 so the associated Event Trigger Logic 1 (ETL1) is used. It is configured to:

- Select input ERU_1A1 as the only input; input Bx is not required because it is not used.
- Enable falling edge detection.
- Select the status flag EXICON1.FL as a sticky flag to indicate the occurrence of the falling edge event.
- Enable output pulse trigger to any available Output Gating Unit (OGUy) for interrupt request generation. In this example, OGU0 is selected

```
XMC_ERU_ETL_CONFIG_t button_event_generator_config =
{
  .input = ERU0_ETL1_INPUTA_P2_5,
  .source = XMC_ERU_ETL_SOURCE_A,
  .edge_detection = XMC_ERU_ETL_EDGE_DETECTION_FALLING,
  .status_flag_mode = XMC_ERU_ETL_STATUS_FLAG_MODE_SWCTRL,
  .enable_output_trigger = true,
  .output_trigger_channel = XMC_ERU_ETL_OUTPUT_TRIGGER_CHANNEL0
};
```

To enable the interrupt request generation, OGU0 has to be configured to react on the trigger event from ETL1. The pattern detection and the peripheral trigger features are not used so it is not necessary to define the related data types.

```
XMC_ERU_OGU_CONFIG_t button_event_detection_config =
{
  .enable_pattern_detection = false,
  .service_request = XMC_ERU_OGU_SERVICE_REQUEST_ON_TRIGGER
};
```

5.6.1.2 Initialization

The selected ETL and OGU channels are then initialized with the configurations defined in Section 5.6.1.1. This is done by calling their respective initialization functions in main().

```
XMC_ERU_ETL_Init(ERU0_ETL1, &button_event_generator_config);
XMC_ERU_OGU_Init(ERU0_OGU0, &button_event_detection_config);
```

Besides the ERU, the ports and the NVIC also need to be initialized to enable the P2.5 input and P0.0 general purpose output pins, and the ERU0.SR0 interrupt.

5.6.1.3 Implementation

When a falling edge is applied on the P2.5 input, the ERU0.SR0 interrupt is triggered. In the interrupt service routine, the status flag is cleared and the P0.0 output is toggled.

```
void ERU0_0_IRQHandler(void)
{
        XMC_ERU_ETL_ClearStatusFlag(ERU0_ETL1);
        XMC_GPIO_ToggleOutput(LED);
}
```

6 Event Trigger Pulse and Level Status Generation

An Event can be represented by a Rising or Falling Edge transition (or by both transitions) of an Event Request carrier signal, indicating that a certain event has occurred. The event edge is regarded as a true event edge if it equals the considered event edge, expected from a selected source according to the Top-Level agenda.

6.1 Event Detection and Evaluation Process

The Event Trigger Logic input signal from the Event Request Select stage represents a certain selected I/O pin or system unit status, hard wired to the ERU input channel x (x=0-3) stage. The Event Trigger Logic in turn detects and sorts the true and false event edge signals by demultiplexing onto two separate branches (see Figure 10).

6.2 Distribution of Detected Events Back to System

- The true event TRIGGER branch can, if enabled, generate an event Trigger Pulse TRx targeting a specific ERU output IOUTy. The channel (y=0-3) is Top-Level selected via the successive Cross Connect Select.
- The true event LEVEL branch sets the Event Flag FLx used by all Output Gating Units for Pattern Detection.

Event Trigger Pulse and Level Status Generation

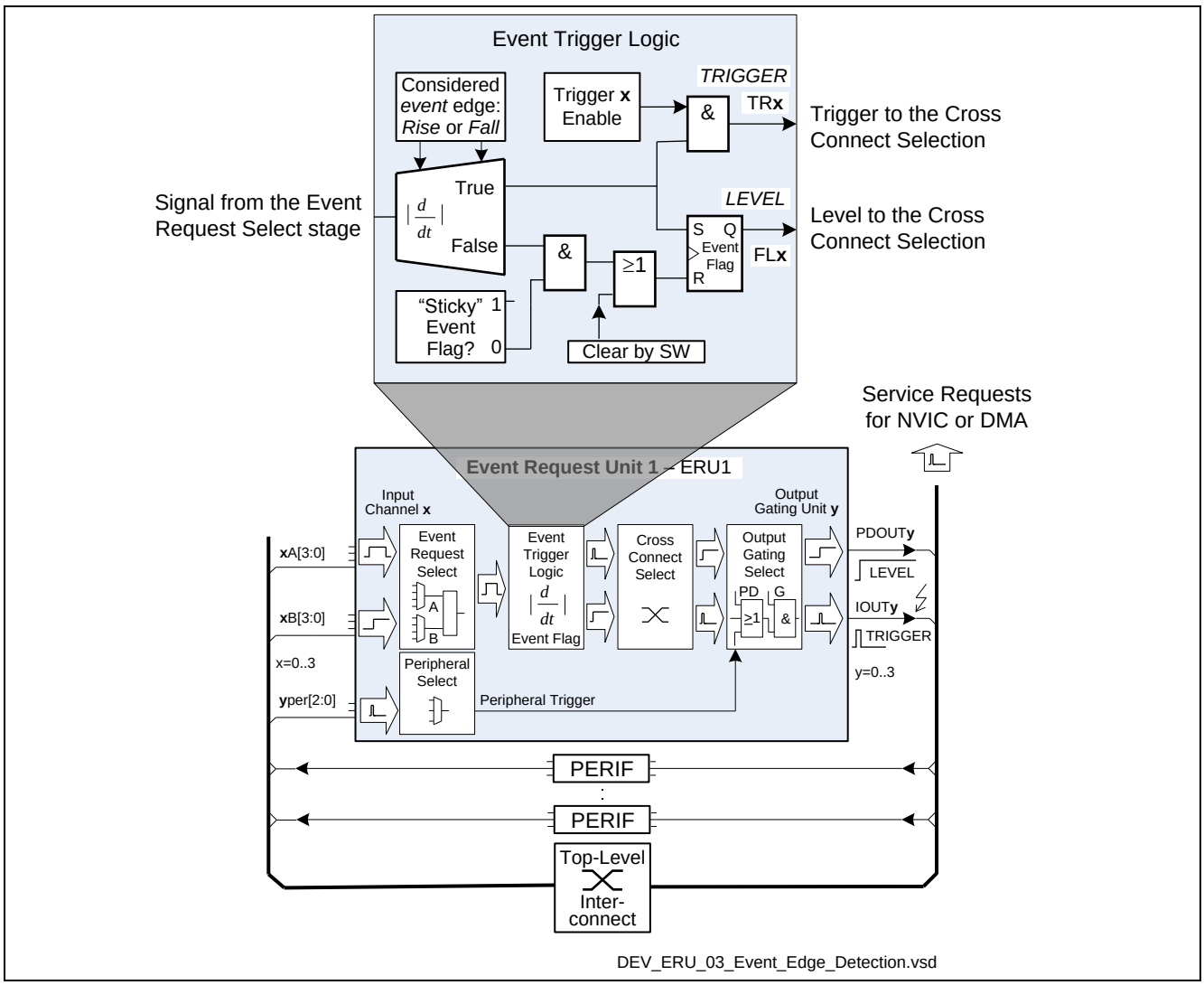

Figure 10 The Event Trigger Logic for Event Detection, Trigger Pulse and Level Status Generation

7 Getting Started with Event Request Unit Trigger Logic

There is one Input Selection Register and two Control Registers per input channel x (x=0-3). These are used to setup the entire functionality of an ERU at the Top Level and to target the Output Channels y (y=0-3):

- EXISEL Input Selection Register
- EXICONx Input and Trigger Control Register
- EXOCONx Output Control Register

The Trigger Pulse TRx (TRIGGER) Enable and the Event Flag FLx (LEVEL) status are controlled by the input registers ERU0_EXICONx and ERU1_EXICONx respectively.

7.1 Trigger Control Setup Example

Initializations and functions are determined by writing to the respective register EXICONx.bitfield (see Figure 11).

Assume the following case:

```
// Initialize
FL = 0; // Clear the event flag, i.e. it will not be rebuild by HW, but by SW
// Select Event Trigger Logic conditions for trigger, level detect and event edge
PE = 1; // set trigger pulse enable, i.e. an output trigger pulse will be created
LD = 0; // define event flag sticky, i.e. it will not be rebuild by HW, but by SW
RE = 1; // detection on Rising Edge, i.e. event edge on positive signal transition
FE = 0; // no detection on Falling Edge
```

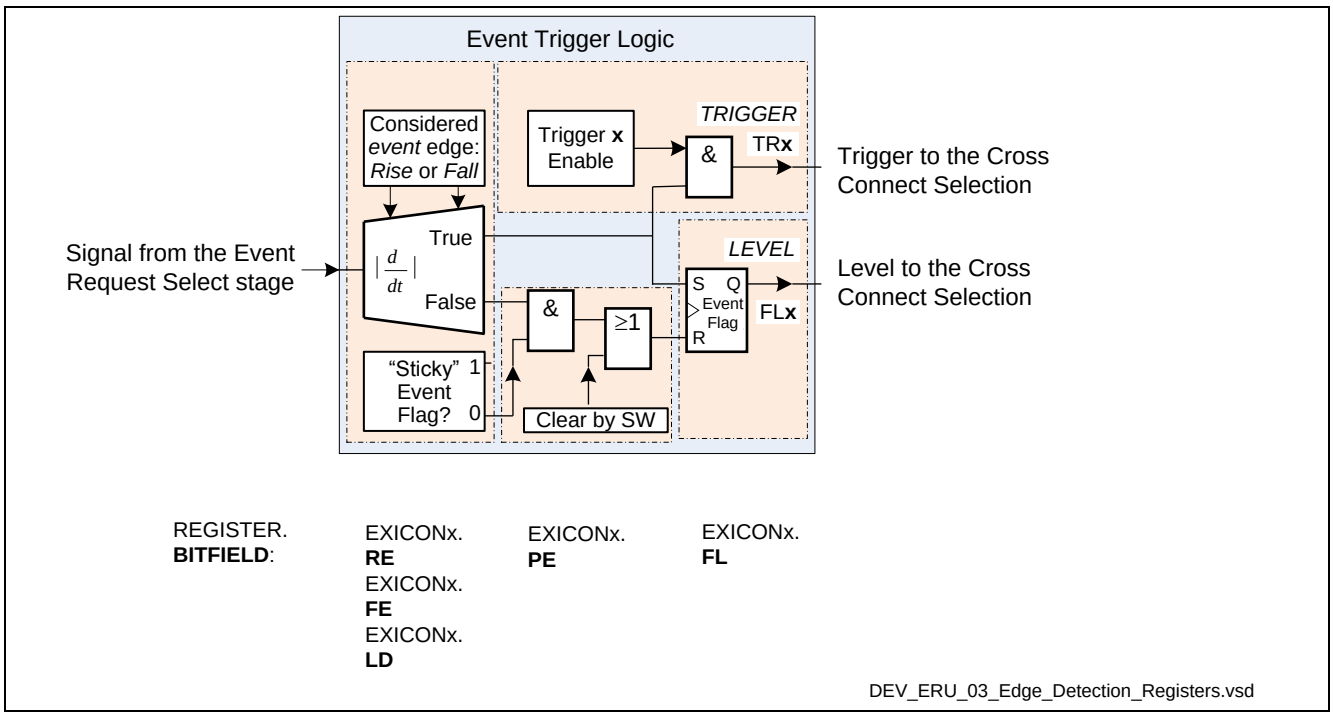

Figure 11 The Event Trigger Logic Input and Control Register

8 Conditional Event Request Handling

An operation that is triggered beyond its deadline should, from a hard realtime perspective, be considered as useless. Such an event might in some applications cause hazards or catastrophic consequences. An ERU offers the possibility to ensure realtime correctness or safety by conditional request handling of compound events.

8.1 Event Request Selection

Each input line, to xA(3:0) or xB(3:0) of an ERU input channel x (x=0-3), is hard wired to a unique event source. This means a specific Event Request, from an I/O pin or a peripheral unit, is selected by a MUX switch at either the xA(3:0) or at the xB(3:0) input line group of the ERU0 or ERU1, according to the ERU Pin Connections tables.

8.2 Signal Combination Logic

An Event Request signal pair (A, B), selected by the input line groups xA(3:0) and xB(3:0) respectively, can represent a considered compound event. Such a conditional signal can be created by the Combination Logic in the channel x input stage before it is transfered to the Event Trigger Logic stage for event edge detection.

8.3 Top-Level Control of Event Request Selection and Combination

Figure 12 and Figure 13 show the implementation of the request selection and combination stages. Event request signals on the input lines xA(3:0) and xB(3:0) of a channel x (x=0-3) can be selected and combined on a Top-Level by the EXISEL and EXICONx registers:

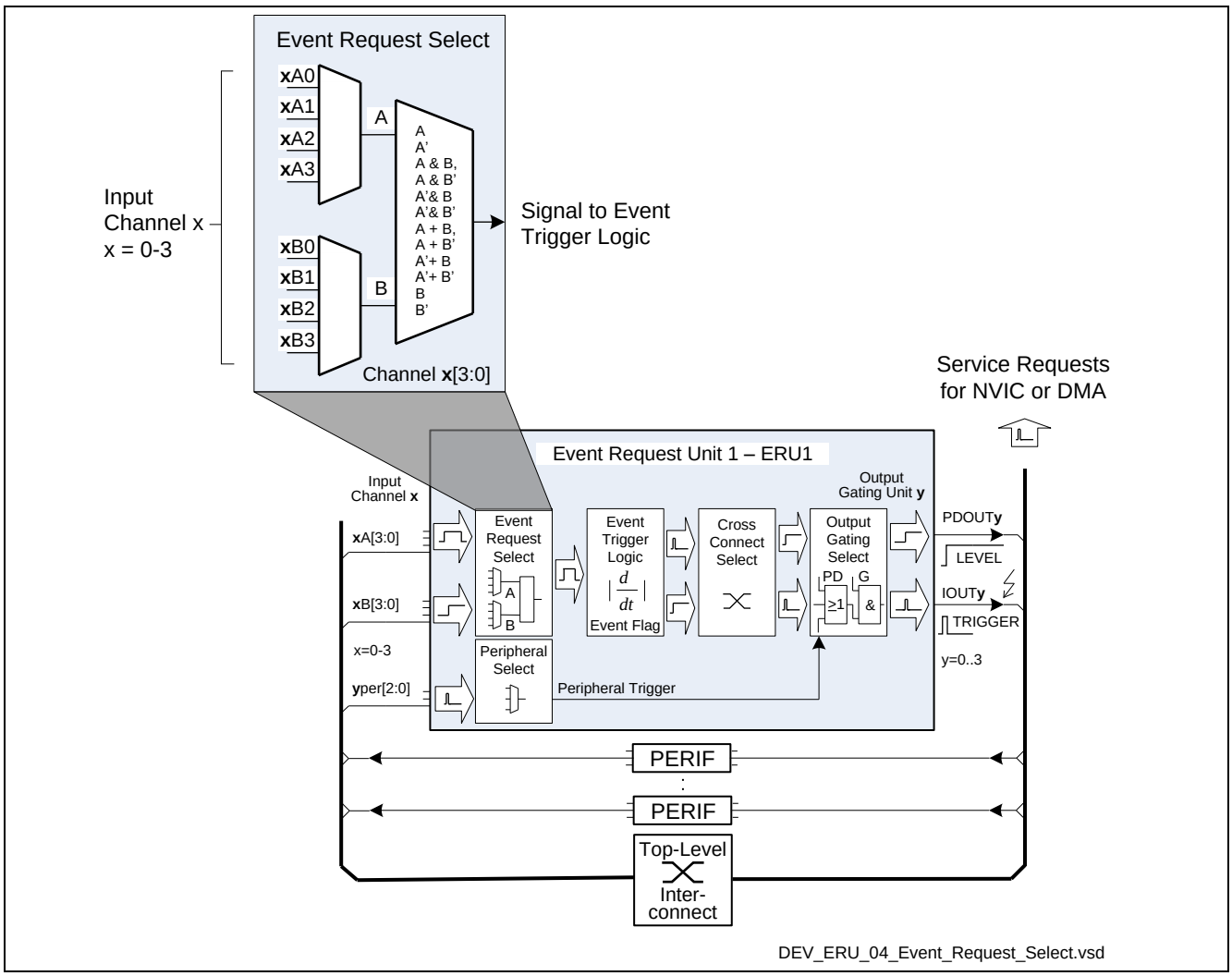

Figure 12 Focusing the ERU1 Event Request Select & Combination Logic Input Stage

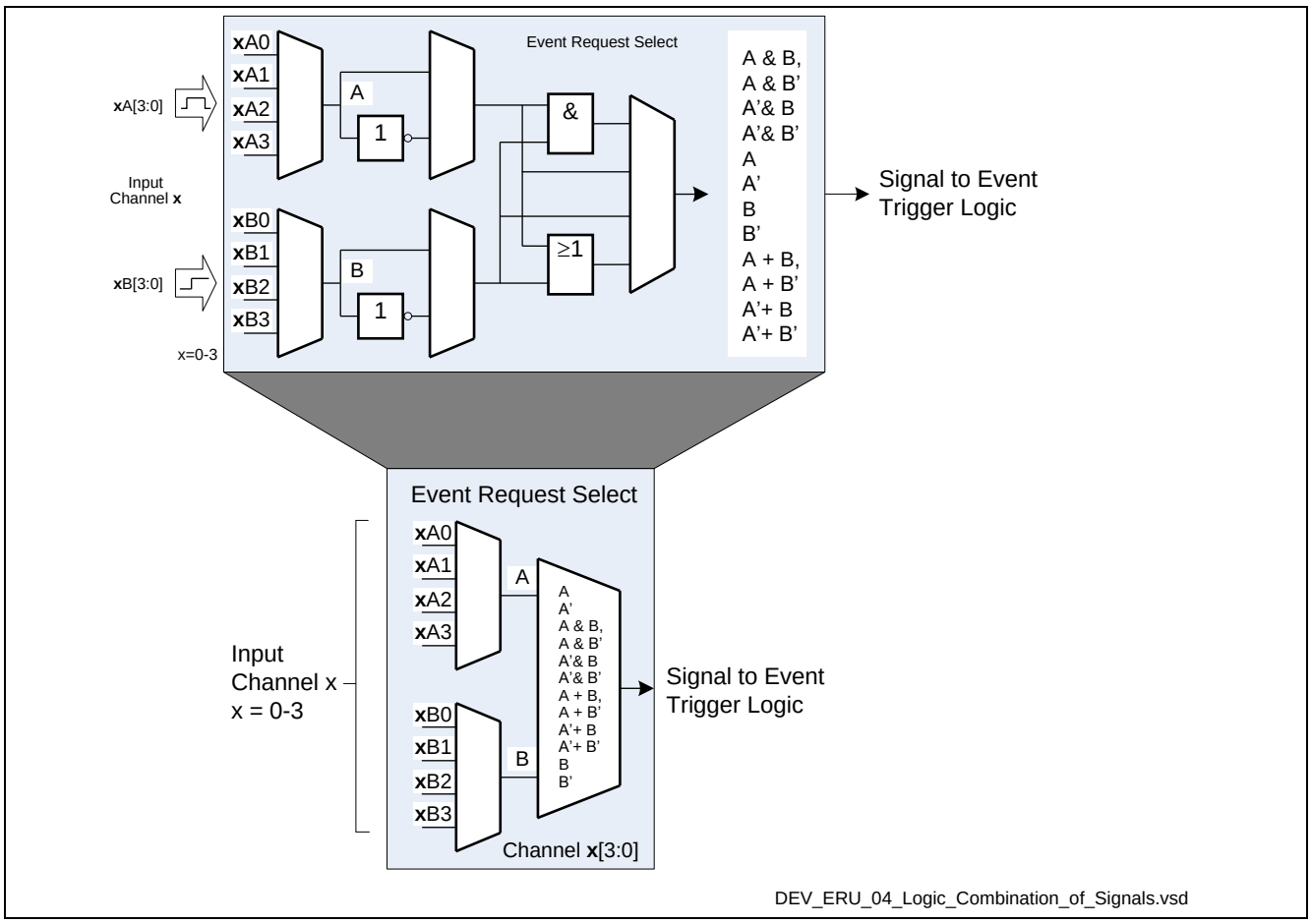

Figure 13 Detailed view of the Event Request Select Logic Stage

9 Getting Started with ERU and Input Selection & Combination

There is one Input Selection Register and two Control Registers per input channel x (x=0-3). These are used to setup the entire functionality of an ERU at the Top Level and to target the Output Channels y (y=0-3):

- EXISEL Input Selection Register
- EXICONx Input and Trigger Control Register
- EXOCONx Output Control Register

9.1 Initialization example

```
// Assume that an EXTERNAL INPUT SIGNAL should be detected within a TIME WINDOW:

// Set up ERU1 Input Channel x=0 for connection to the EXTERNAL INPUT SIGNAL at
// Port Pin P1.15 and CAPCOM8 CCU80.ST0 Status Bit controlling the TIME WINDOW.

// Select Event Request for input A and B in the ERU1_EXISEL register bitfields:
EXS0A = 0; // select input ERU1_0A0 - i.e. A connected to GPIO port pin P1.5
EXS0B = 1; // select input ERU1_0B1 - i.e. B connected to CCU80.ST0 status bit 0

// Select logical combinations that should be taken into account as event request
NA = 0; // input A is used directly, i.e. not inverted
NB = 0; // input A is used directly, i.e. not inverted
SS = 3; // setup source combination, i.e. logical condition: input A AND input B
```

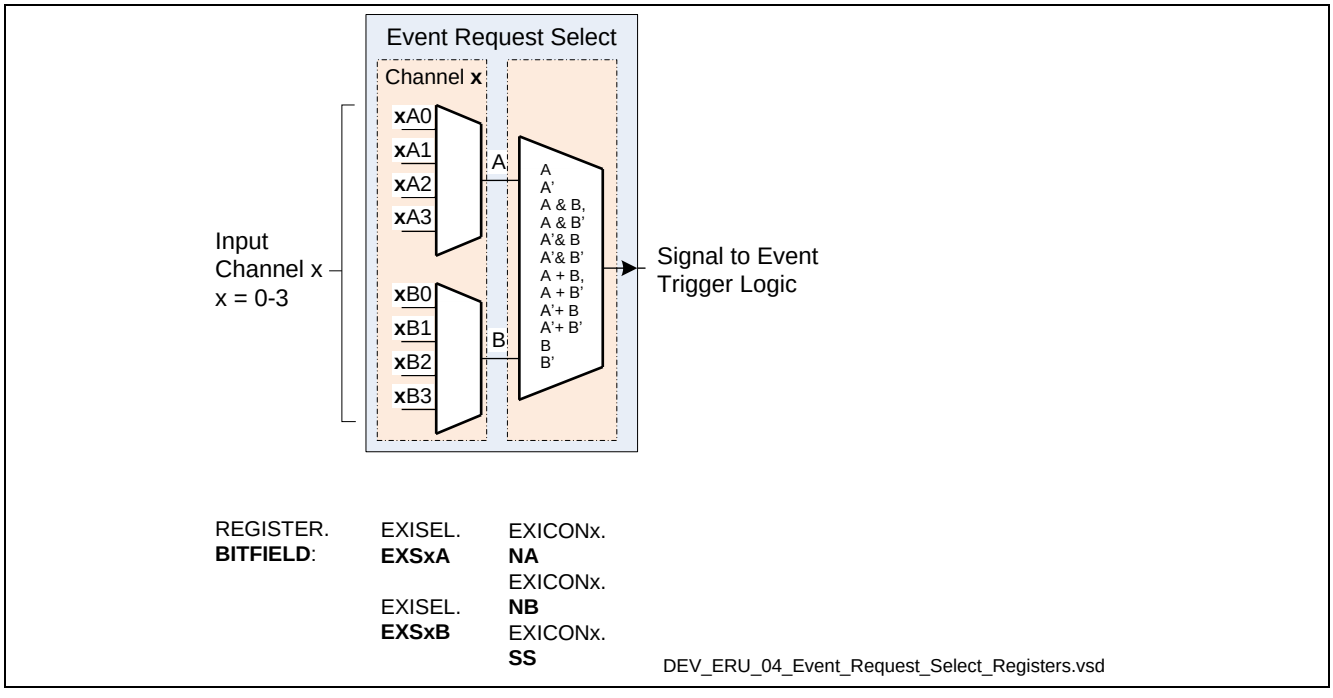

Figure 14 The Event Request Select Stage Control Registers

10 Runtime Handling of ERU and Input Selection & Combination

10.1 Handling Event Concatenation and using Pattern Match Detection

The following example shows a state-machine-like configuration setup of ERU1. It is used to explain how a certain sequence of events can be concatenated on a Top-Level control and detected by an Event Pattern Match on the ERU output gating stage. One example scenario is a Debounce Rejection Filter.

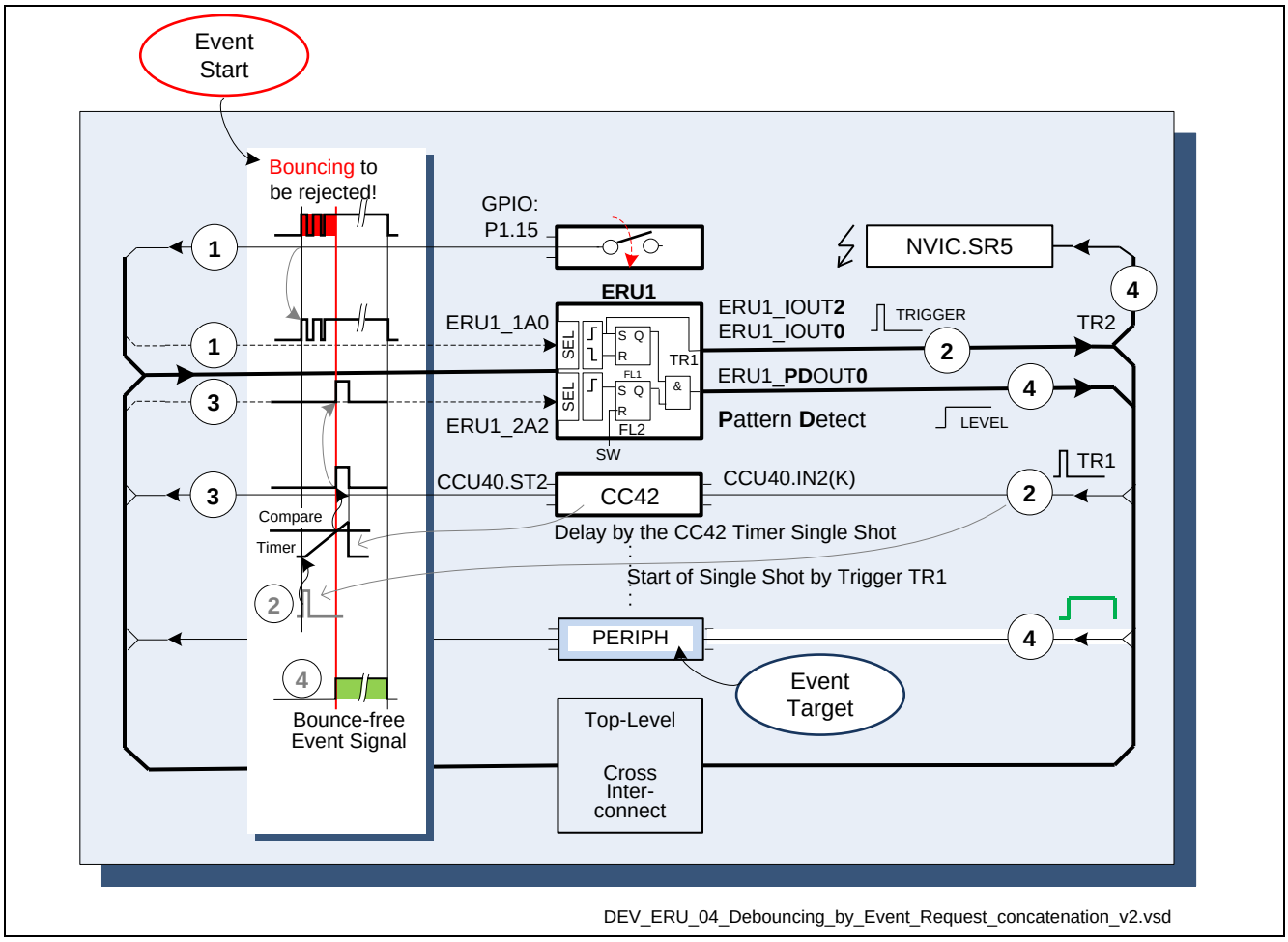

Figure 15 Concatenation of events with Time Window for a press-button Debounce Rejection Filter

1. Bouncing signal from SWITCH-ON. This signal is connected to the ERU1 Channel 1 Input Selection Pin A0 (ERU1_1A0). The event is detected on the Rising Edge, memorized in event flag FL1 and then trigger pulse TR1 is generated on the Output Gating Unit 2 (IOUT2) after the ERU1 internal Cross Connect Stage.
2. Trigger TR1 on Output Gating Unit 2 (IOUT2) starts the Timer CCU40T42 in Single Shot mode by an External Event Control command (including the Extended Mode condition for *Flush/Start*).
3. On the single shot timer Compare Event that starts the Single Shot time frame, the CCU40ST2 status bit ST2 is set. This signal is connected to the ERU1 Channel 2 Input Selection Pin A2 (ERU1_2A2) and sets the sticky flag FL2 and sends a trigger pulse TR2 on IOUT0 after the Cross Connect Stage.

4. On the Output Gating Unit 0 (IOUT0) pin the trigger pulse TR2 can be distributed to the Service Request Line nr 5. On the PDOUT0 output pin of the Output Gating Unit 0 there is Pattern Match Detection by the AND-Gate and the flags FL1 and FL2. This is the debounced press-button switch signal that has been extracted by event concatenation. It is then sent to the Peripheral Unit named the “Event Target”.

11 Time Windowing

11.1 The ERU and Sequential Events Control

The ERU top-level control of events enables creation of comprehensive embedded tasks in HW. These tasks can concatenate actions from different units and so act like a state machine. This enables interactions to be controlled at a low level without involving SW. The results are controlled by the event flow and an action scheme that is configured on a top-level by the user.

11.2 Use Case with the ERU in Time Windowing

An Event Request signal pair (A,B), selected by the input line groups xA(3:0) and xB(3:0) respectively, can represent a considered compound event (see Figure 16). Such a conditional signal can be derived by the Combination Logic in the channel x input stage. When using timers as input sources, complex windowing functions can be realized.

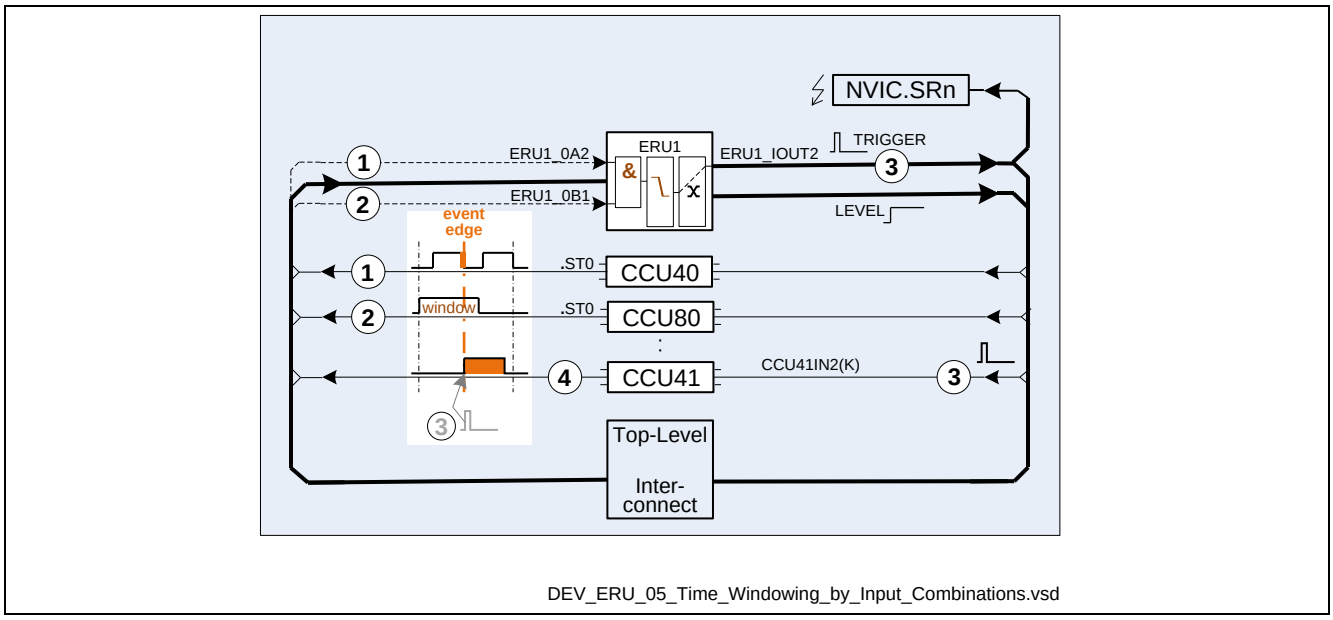

Figure 16 Time Windowing by Input Combination of Timer Signals

11.3 Using Delayed Events in External Events Control as a State-Machine

The ERU Output Gating Stage OGUy (y=0-3) offers the possibility to construct compound event conditions that involve memorized events and require a certain sequence of events to be satisfied. The event flags FLx, set by “events before”, can be used for gating conditions of the “events after” that occur at the input channels later on.

11.3.1 Combining Event Flags with Input Channel Events

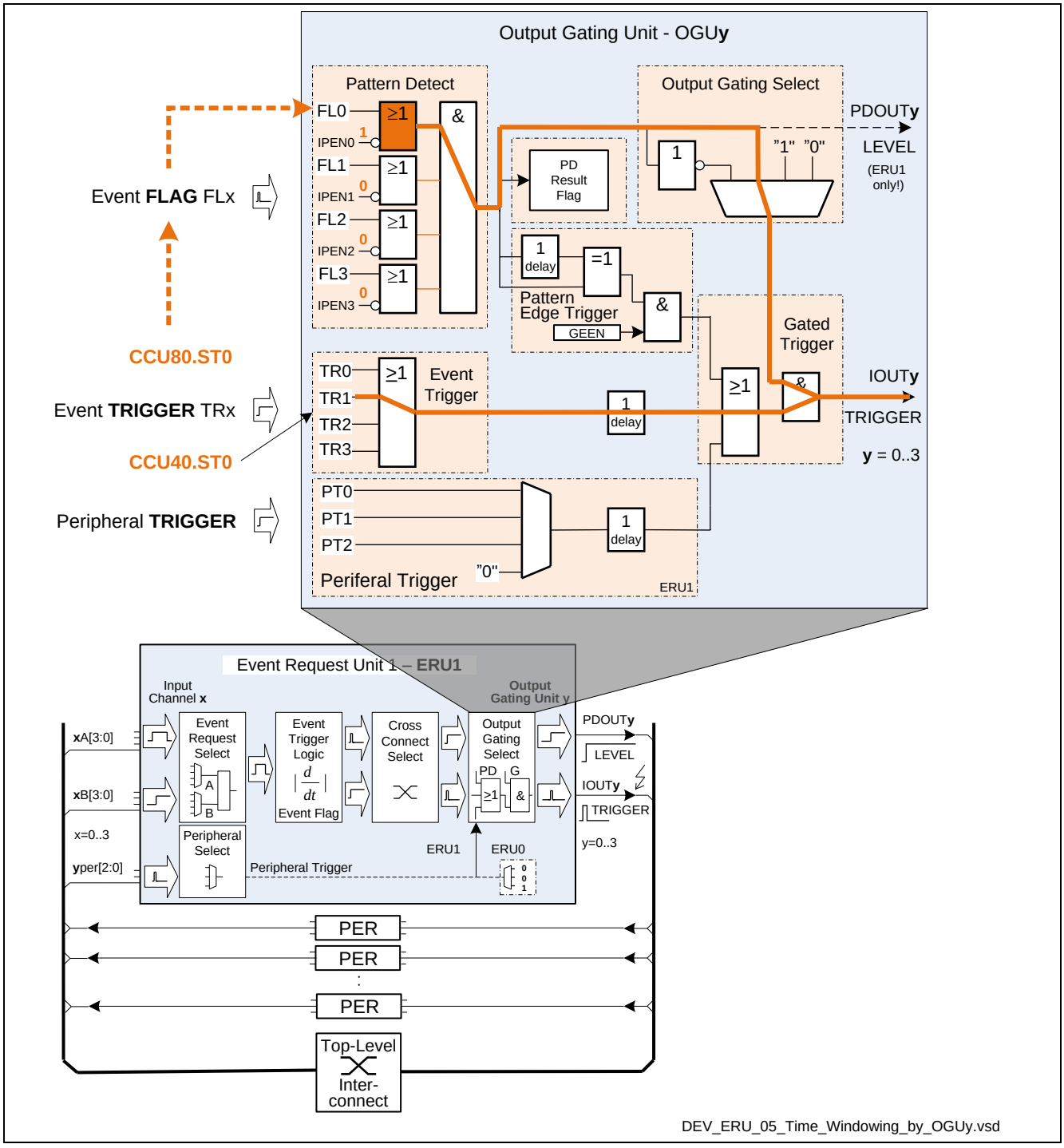

Figure 17 Combining Event Flags and External Events Control

11.3.2 Combining Event Flags with the Designated Peripheral Trigger Inputs

ERU1 has designated trigger inputs from the ADC and from some of the CAPCOM4/-8 units that can be linked directly to the Gated Trigger stage of an Output Gating Unit channel OGUy (see Figure 18). By this “kitchen

entrance”-like concept these Peripheral Triggers act immediately, unconditionally in the output gating by the event flags FLx.

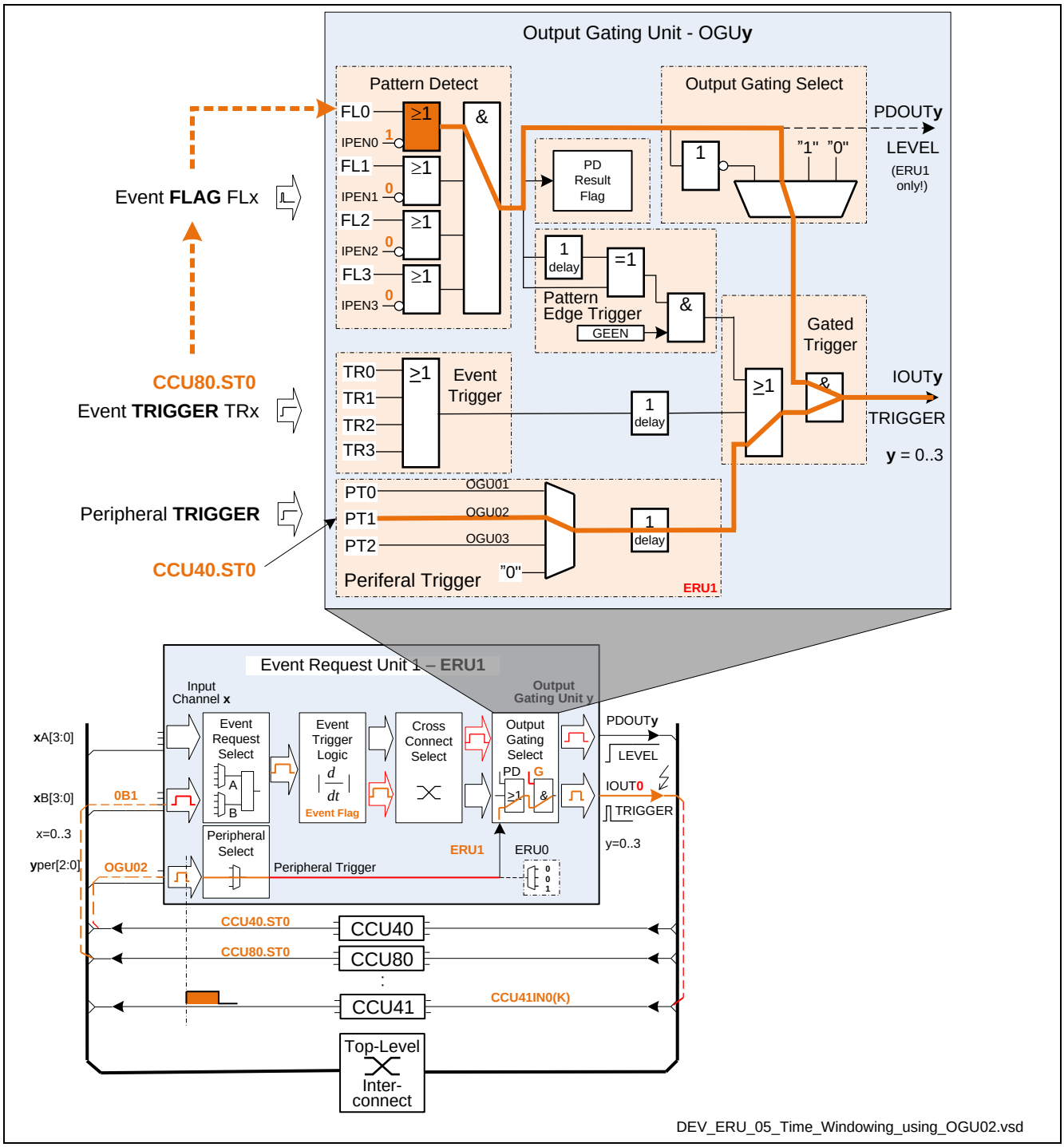

Figure 18 Combining Event Flags and Peripheral Trigger

12 Event Request Routing to Service Providers

12.1 The Cross Connect Selection

The Cross Connect Select stage, for each channel x of an ERU, is an Event Trigger pulse routing matrix that is able to switch the trigger pulse distribution path back to the system via any output channel y (y=0-3). The target can be a certain event service action provider that is defined by the Top-Level control by the ERU Pin Connections tables.

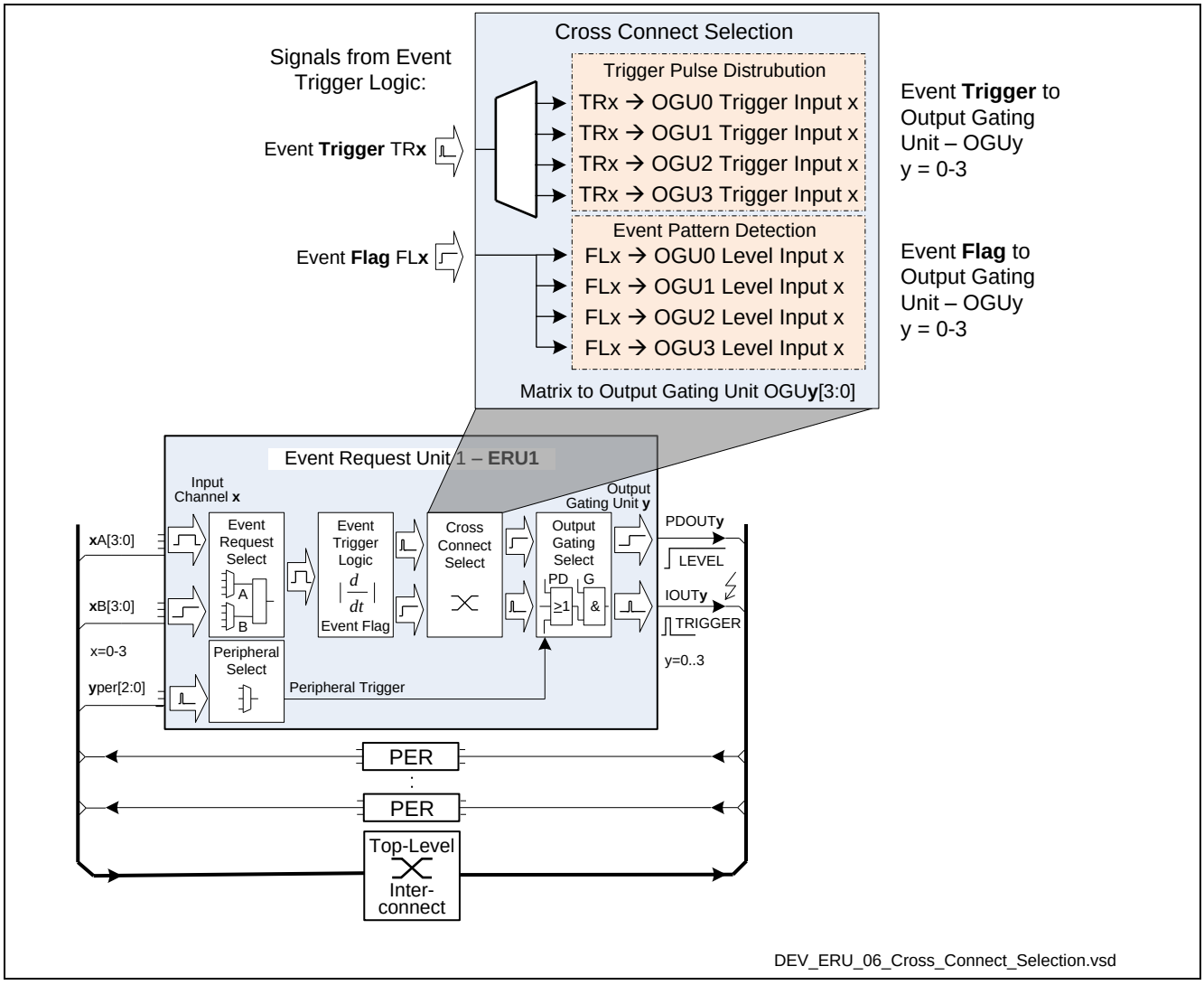

Figure 19 The Cross Connect Selection Stage

12.2 Top-Level Interconnect Matrix

Most interconnect lines go directly from one system unit to another unit, without passing an ERU. For example, an ADC conversion can be started directly by a signal from a timer. These immediate interconnections are described virtually by the so called Top-Level Interconnect matrix - a table which is embodied by a block (see Figure 19).

12.3 The ERU Pin Connection Tables

It should be mentioned and understood though that the ERU Pin Connections tables (which describe the ERU0 and ERU1 interfaces to peripheral units or IOs) are actually subsets of the total system Top-Level Interconnect matrix. They are split up like this in order to ease understanding of those paths where an ERU is involved.

13 Getting Started with Event Request Unit Cross Connect

There is one Input Selection Register and two Control Registers per input channel x (x=0-3). These are used to setup the entire functionality of an ERU at the Top Level and to target the Output Channels y (y=0-3):

- EXISEL Input Selection Register
- EXICONx Input and Trigger Control Register
- EXOCONx Output Control Register

The Cross Connect Selection should be mapped to the EXICONx register bit field OCS (see Figure 20).

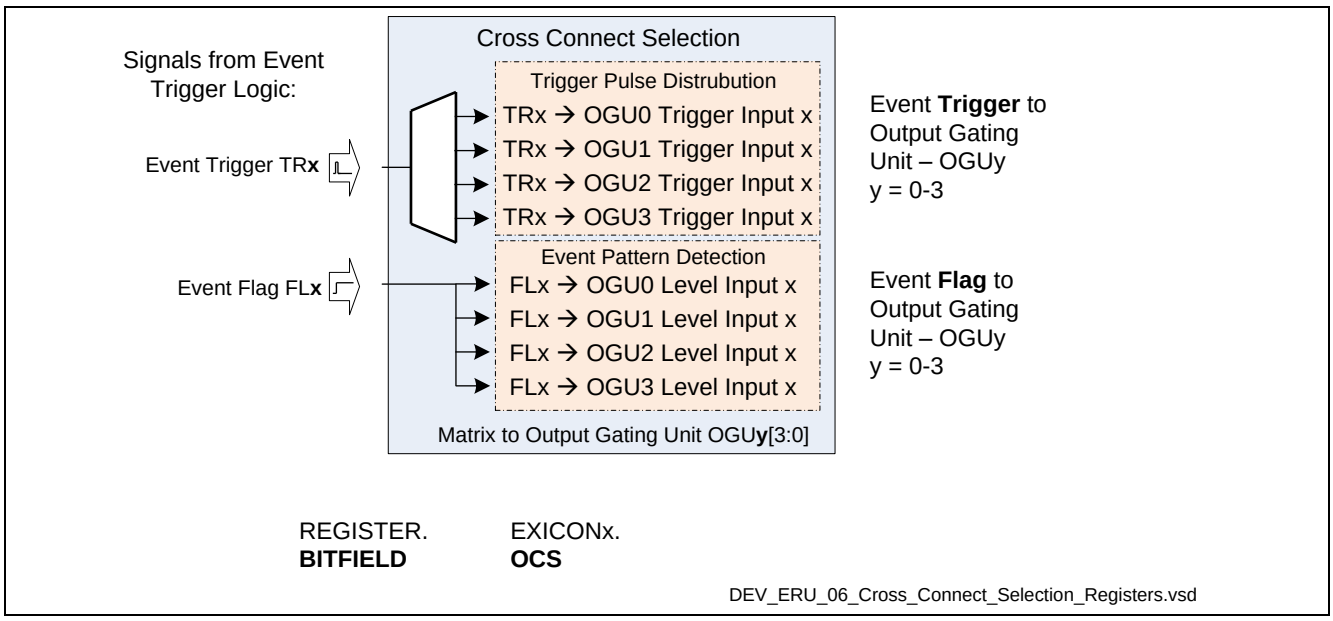

Figure 20 Using the Input and Trigger Control Register EXICONx for Cross Connect Selection

```
// Pseudo code Example:
// Perform Cross Connect Selection, e.g. select target output channel y = 2
// in the 3-bit bit field OCS of the register ERU1_EXICON0, to map
// a Cross Connection for the Event Trigger Pulse TR0 in channel x =0:
EXICON.OCS = 010B;
// The channel output IOUT2 would trigger the following destinations in
// this case:
   // CCU4x.IN2(K)    - i.e. CAPCOM4 Unit CCU4x, Slice CC42 Timer Input
   // CCU8x.IN2(G)    - i.e. CAPCOM8 Unit CCU8x, Slice CC42 Timer Input
   // VADC.G2REQTRN   - i.e. ADC Trigger Request N Input, Group 2
   // VADC.G3REQTRN   - i.e. ADC Trigger Request N Input, Group 3
   // ERU1.1B3 - i.e. A trigger feed-back to ERU1 Channel Input 1B3
   // NVIC.SR7 - i.e. A trigger puls to Service Request Line nr 7
   // POSIF0.MSET(F) - i.e. POSIF 0 Multi Channel Next Pattern Update Set
   // POSIF1.MSET(F) - i.e. POSIF 1 Multi Channel Next Pattern Update Set
   // according to the ERU1 Pin Connection Tables.
```

14 Top-Level Control of External Event Requests

Top-Level Control of Event Requests enables flexible control and cross-unit distribution of event signals. The Event Request Unit (ERU1) can combine events via the Top-Level Interconnect Matrix. This means cross-unit event flow can be controlled at the top-level according to a user request-to-action event pattern setup. This can be then gated without requiring any SW interactions (see Figure 21).

14.1 Event Request Selection

Each input line, to xA(3:0) or xB(3:0) of an ERU input channel x (x=0-3), is hard wired to a specific event source. So, a specific Event Request, from an I/O pin or a peripheral unit, is selected by a MUX switch at either the xA(3:0) or at the xB(3:0) input line group of the ERU0 or ERU1, according to the ERU Pin Connections tables.

14.2 Signal Combination Logic

An Event Request signal pair (A,B), selected by the input line groups xA(3:0) and xB(3:0) respectively, can represent a considered compound event. Such a conditional signal can be created by the Combination Logic of the channel x input stage before it is transferred to the Event Trigger Logic stage for event edge detection.

14.3 Input and Trigger functions

A signal edge is regarded as a true event edge if it complies with the considered edge, positive or negative. If it is a true event edge an event flag FLx is set and a trigger pulse TRx is created. If enabled, the trigger pulse and the event flag level status are passed to the Cross Connect Select stage. The flag can be reset by a false event or by SW.

14.4 Cross Connect Selection

The Cross Connect Select stage for each channel x of an ERU is an Event Trigger pulse routing matrix. The matrix is able to switch the trigger pulse distribution path back to the system via any output channel y (y=0-3). It can target a certain event service action provider according to the Top-Level control by the ERU Pin Connections tables.

14.5 Output Gating Selection

The Output Gating Selection stage of an ERU can gate the Trigger Pulse signals (TRIGGER) in the output channels IOUTy (y=0-3) by the Event Pattern Match/Mismatch Detect status (LEVEL) output signal, PDOUTy (y=0-3). An Event Pattern is a combination of memorized x-channel events stored in the event flags FLx (x=0-3).

14.6 Top-Level Interconnect Matrix

Most interconnect lines go directly from one system unit to another unit without passing an ERU. For example, an ADC conversion can be started directly by a signal from a timer. These immediate

interconnections are described virtually by the so called Top-Level Interconnect matrix. This table is embodied by a block in the drawing (see Figure 21).

14.6.1 The ERU Pin Connection Tables

It should be mentioned and understood though that the ERU Pin Connections tables (which describe the ERU0 and ERU1 interfaces to peripheral units or IOs) are actually subsets of the total system Top-Level Interconnect matrix. They are split up like this in order to ease understanding of those paths where an ERU is involved.

Top-Level Control of External Event Requests

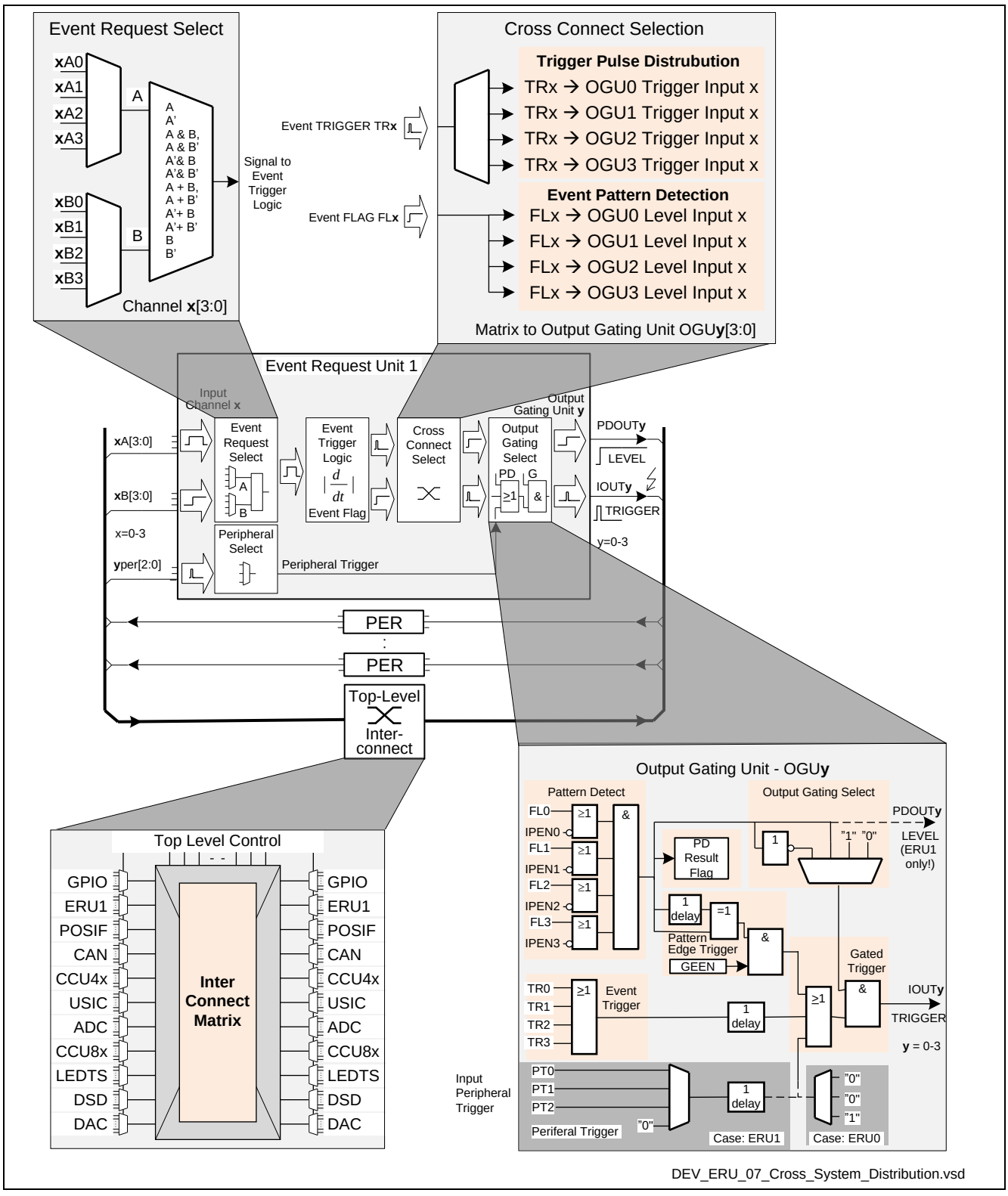

Figure 21 Cross System Events Top-Level Control and Distribution

15 Getting Started with Event Request Unit and Top-Level Control

There is one Input Selection Register and two Control Registers per input channel x (x=0-3). These are used to setup the entire functionality of an ERU at the Top Level and to target the Output Channels y (y=0-3):

- EXISEL Input Selection Register
- EXICONx Input and Trigger Control Register
- EXOCONx Output Control Register

Initializations and functions are determined by writing to the respective register bit fields.

15.1 Cross Interconnection Example – Using CCU4, ERU1, ADC and GPIO

15.1.1 ADC Request on Port Pin P2.1 State AND the Timer CCU40CC40 Status Bit

Setup ERU1 Input Channel x=0, targeting the Output Channel y=2 (IOUT2), from which a link to the ADC Trigger Request N Input, Group 2 or 3 can be routed – as follows:

```
// Select Event Request for input A and B in the ERU1_EXISEL register bit fields:
EXS0A = 2; // select input ERU1_0A2 – i.e. A connected to CCU40.ST0 status bit 0
EXS0B = 0; // select input ERU1_0B0 – i.e. B connected to a GPIO port pin P2.1

// Select logical combinations that should be taken into account as event request
NA = 0; // input A is used directly, i.e. not inverted
NB = 0; // input B is used directly, i.e. not inverted
SS = 3; // setup source combination, i.e. logical condition: input A AND input B

// Select Event Trigger Logic conditions for trigger, level detect and event edge
PE = 1; // set trigger pulse enable, i.e. an output trigger pulse will be created
LD = 0; // define event flag sticky, i.e. it will not be rebuild by HW, but by SW
RE = 1; // detection on Rising Edge, i.e. event edge on positive signal transition
FE = 0; // no detection on Falling Edge

// Perform Cross Connect Selection, i.e. select target output channel y (here y=2)
// for the Event Trigger Logic Output Pulse TR0 from input channel x (here x=0):
// Target register for the following mapping is EXICONx, i.e. here ERU1_EXICON0:
OCS = 2;
     // CCU4x.IN2(K)  - i.e. CAPCOM4 Unit CCU4x, Slice CC42 Timer Input
     // CCU8x.IN2(G)  - i.e. CAPCOM8 Unit CCU8x, Slice CC42 Timer Input
     // VADC.G2REQTRN - i.e. ADC Trigger Request N Input, Group 2
```

```
      // VADC.G3REQTRN- i.e. ADC Trigger Request N Input, Group 3
      // ERU1.1B3 - i.e. A trigger feed-back to ERU1 Channel Input 1B3
      // NVIC.SR7 - i.e. A trigger puls to Service Request Line nr 7
      // POSIF0.MSET(F) - i.e. POSIF 0 Multi Channel Next Pattern Update Set
      // POSIF1.MSET(F - i.e. POSIF 1 Multi Channel Next Pattern Update Set

// Select Event Output Trigger Control 2 Register ERU1_EXOCON2 for output gating
// Select Output Gating functions – Here: just a straight forward alternative:
// Target register for the following mapping is EX0CONy, i.e. here ERU1_EXOCON2:
GEEN = 0; // disable "Gating Event Enable on Pattern Detection Changes" trigger
GP   = 1; // set "Output Gating Select on Pattern Detection" =1, to activate IOUT2
IPEN0, IPEN1, IPEN2, IPEN3 = 0; // disable the Pattern Detection Enable flags.
```

16 Control by Event Pattern Match

An Output Gating Unit OGUy (y=0-3) can gate all the event signals of an ERU into one output trigger signal (TRIGGER) at the IOUTy output pin - controlled by an Event Pattern Detect status (LEVEL) at the PDOUTy pin. An Event Pattern is a combination of true events, selected by an Input Enable IPENx bit for each event flag FLx.

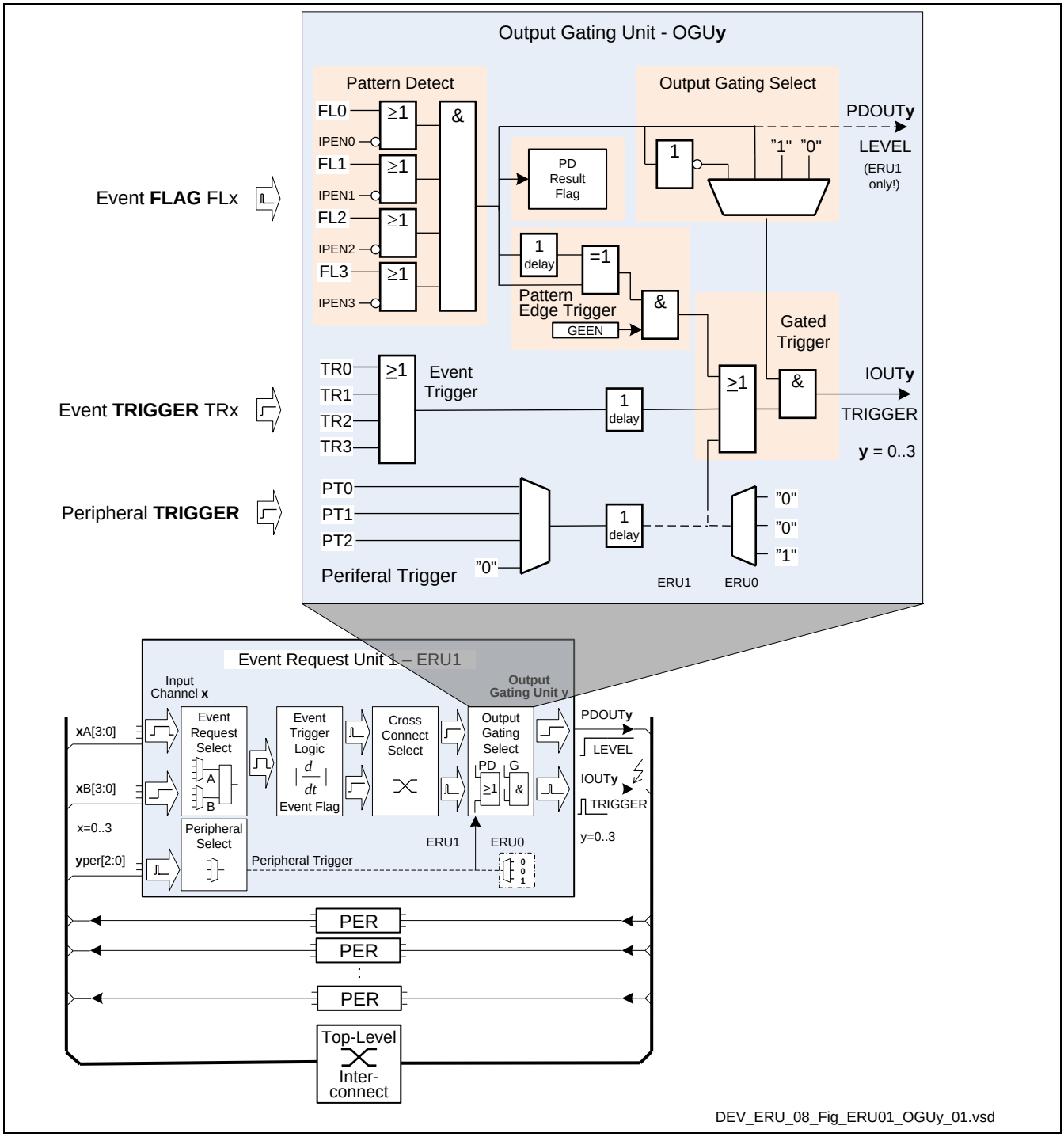

Figure 22 ERU and the Output Gating Unit (OGUy, y=0-3)

16.1 The Output Gating Unit in Detail

The Output Gating Unit OGUy (y=0-3) is comprised of three stages:

- a Pattern Detect stage
- a Pattern Edge Trigger stage
- and an Output Gating Select MUX that controls the Gated Trigger stage (see Figure 23).

Only the enabled inputs of event flags FLx values satisfy the pattern that can gate ERU trigger events as a “TRIGGER” from the IOUTy output pin.

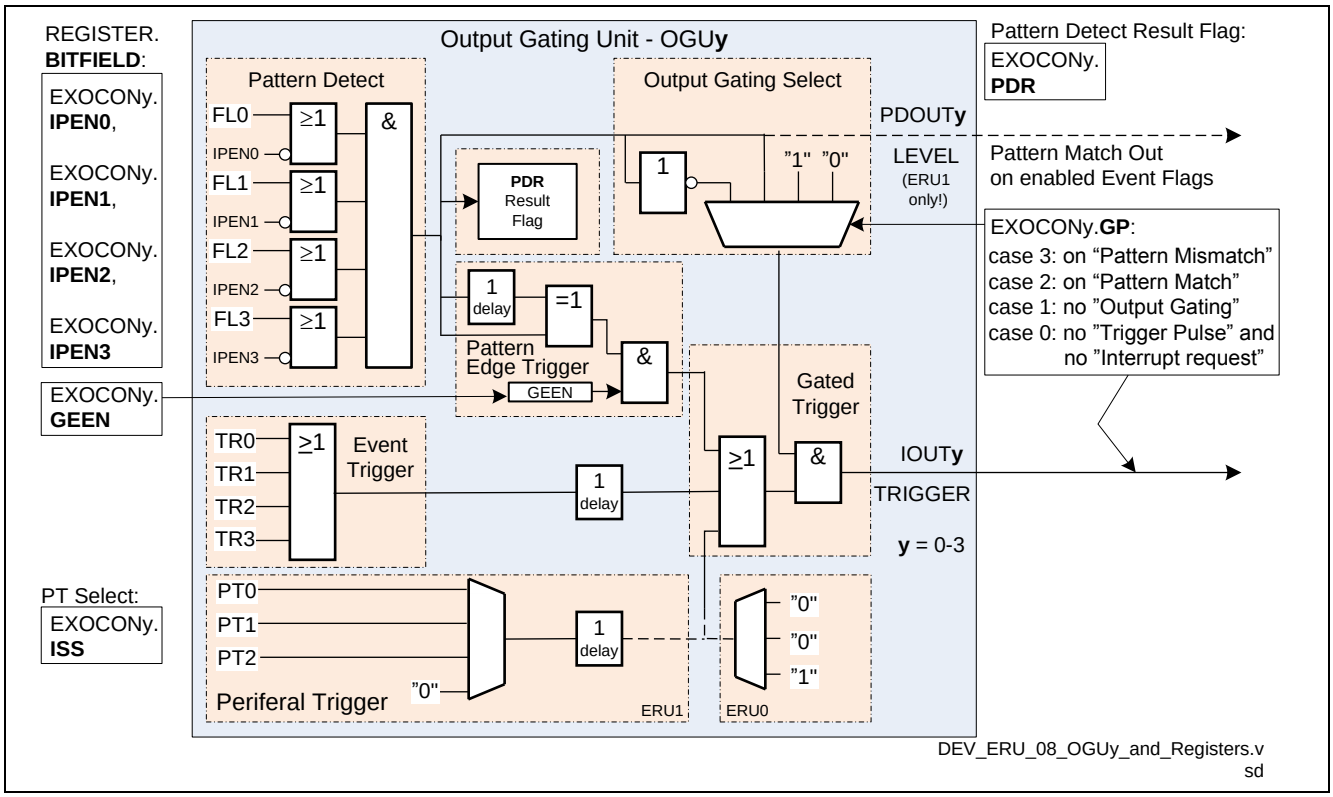

Figure 23 The Output Gating Unit (OGUy) in Detail

16.1.1 Pattern Detect status – PDOUT

The Pattern Detect function is an AND-operation of all event memorizing flags FLx that are regarded by the Pattern Enable EXOCONy.IPENx bits. The PD Result flag is affected and the status is linked to the Output Gating Select MUX – and (in ERU1 only) also delivered as a “LEVEL” event to the PDOUTy output pin.

16.1.2 Pattern Event Edge Detection

If at least one pattern detection input flag FLx is enabled (by IPENx) and a change of the pattern detection result from pattern match to pattern miss (or vice-versa) is detected, then a trigger event is generated to indicate a pattern edge trigger event (if enabled by EXOCONy.GEEN) and linked to the Gated Trigger stage at the IOUTy.

16.1.3 Output Gating Selection

The Output Gating condition to be selected by the EXOCONy.GP case switch (MUX) is one of the following:

- case 3: on Pattern Mismatch
- case 2: on Pattern Match
- case 1: no Output Gating
- case 0: no Trigger Pulse (nor any Interrupt Request)

Note: The Pattern Detect status (LEVEL) is distributed in any case via the PDOUTy pin. (ERU1 only).

16.1.4 Trigger and Interrupt Output – IOUT

The Trigger and Interrupt pin IOUTy is the derived OR combination of three trigger categories (which is gated according to the Output Gating condition selection that is mapped to the EXOCONy.GP register bit field):

- A Pattern-Match / Pattern-Miss Event Edge Trigger
- Any combination of the Event Edge Triggers TR0, TR1, TR2 or TR3, detected by the ERU input channels x
- A Peripheral Trigger (ERU1 only though) that is bypassed directly to the OGUy (via the “Kitchen Entrance”)

16.2 Use Case Example 2: Gating a Peripheral Trigger Event with a Port Pin

In this use case example, the CCU40 timer slice CC41 is running in the edge aligned mode and the period match while counting up event generates an interrupt request only if it occurs after a falling edge on the port pin P2.5. This can be realized by using ERU0 to gate the CCU40 peripheral trigger depending on the pattern detection status on P2.5.

The example is developed based on the XMC1200 device.

16.2.1 XMC Lib Implementation

The next sections describe how the Infineon XMC library (XMC Lib) can be used to implement the example.

16.2.1.1 Configuration

Since P2.5 is connected to the channel 1 input A1, the associated Event Trigger Logic 1 (ETL1) is used. It is configured to:

- Select input ERU_1A1 as the only input; input Bx is not required because it is not used.
- Enable falling edge detection.
- Select the status flag EXICON1.FL to be automatically updated with each transition on the channel input.
- Disable the output pulse trigger; the output trigger channel need not be defined because it is not used.

```
XMC_ERU_ETL_CONFIG_t button_event_generator_config =
{
  .input = ERU0_ETL1_INPUTA_P2_5,
  .source = XMC_ERU_ETL_SOURCE_A,
  .edge_detection = XMC_ERU_ETL_EDGE_DETECTION_FALLING,
```

```
  .status_flag_mode = XMC_ERU_ETL_STATUS_FLAG_MODE_HWCTRL,
  .enable_output_trigger = false,
};
```

To gate the CCU40 peripheral trigger with the pattern detection status of the input channel 1, the Output Gating Unit 0 (OGU0) is used and configured to:

- Select CCU40.SR0 as the peripheral trigger input.
- Select the status flag from ETL1 as the pattern detection input.
- Configure the interrupt request generation to be based on the logical AND of the peripheral trigger and pattern match events.

```
XMC_ERU_OGU_CONFIG_t button_event_detection_config =
{ .enable_pattern_detection = false,
  .pattern_detection_input = XMC_ERU_OGU_PATTERN_DETECTION_INPUT1,
  .peripheral_trigger = XMC_ERU_OGU_PERIPHERAL_TRIGGER1,
.service_request = XMC_ERU_OGU_SERVICE_REQUEST_ON_TRIGGER_AND_PATTERN_MATCH
};
```

16.2.1.2 Initialization

The selected ETL and OGU channels are then initialized with the configurations defined in Section 16.2.1.1. This is done by calling their respective initialization functions in main().

```
  XMC_ERU_ETL_Init(ERU0_ETL1, &button_event_generator_config);
  XMC_ERU_OGU_Init(ERU0_OGU0, &button_event_detection_config);
```

Besides the ERU, the following peripherals also need to be initialized:

- Ports to enable the P2.5 input and P0.0 general purpose output pins.
- NVIC to enable the ERU0.SR0 interrupt.
- CCU40 timer slice CC41 to count in edge aligned mode and to generate a service request upon period match while counting up.

16.2.1.3 Implementation

Only period match while counting up events that occur after the falling edge and before the next rising edge on P2.5, generate the interrupt request to the CPU.

In the interrupt service routine, the status flags are cleared and the P0.0 output is toggled.

```
void ERU0_0_IRQHandler(void)
{
  XMC_ERU_ETL_ClearStatusFlag(ERU0_ETL1);
  XMC_CCU4_SLICE_ClearEvent(SLICE_PTR, XMC_CCU4_SLICE_IRQ_ID_PERIOD_MATCH);
  XMC_GPIO_ToggleOutput(LED);
}
```

17 Revision History

Current Version is V1.0, 2015-07

Page or Reference	Description of change
V1.0, 2015-07	
	Initial Version

www.infineon.com

Edition 2015-07

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference

AP32306

Legal Disclaimer

THE INFORMATION GIVEN IN THIS APPLICATION NOTE (INCLUDING BUT NOT LIMITED TO CONTENTS OF REFERENCED WEBSITES) IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office. Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.